APPLE II® COMPATIBLE

# MUSIC SYNTHESIZER

ALF PRODUCTS

The

# Apple Music Synthesizer Owner's Manual

## Complete Instructions
for the
## 10-5-16
## Apple Music Synthesizer

"the amazing thing about a Dancing Bear
is not how well he Dances;
but that he can
Dance At All"

ALF Products Inc.
1448 Estes
Denver, CO 80215
U.S.A.

Part Number 11-1-6B
(Replacing the previous 11-1-6
and 11-1-7.)

The information contained in this manual was believed to be accurate at the time of publication. Although this manual has been carefully checked for accuracy by our inebriated technical staff, we assume no responsibility for errors or omissions. Independent verification of specifications is recommended in cases where this entertainment product is to be used or modified for use in other applications. ALF reserves the right to make changes in the product and/or specifications without notice.

this manual is dedicated to
all those who struggled along
with the previous version

Praise be to Xerox,
creator of the Diablo™ Hytype™;
but All Hope Abandon
ye who try to use the
Word Processing Enhancements
or the
Advanced Functions Groups

Project Engineers: Rick Harman, John Ridges, Philip Tubb.
Software Design: John Ridges, Forrest Thiessen, Philip Tubb.
Manual written by: Philip Tubb.
Manual graphics by: Rick Harman.
Automatic graphics by: John Ridges, Philip Tubb.
And 5 proofreaders who shall, for obvious reasons, remain nameless.

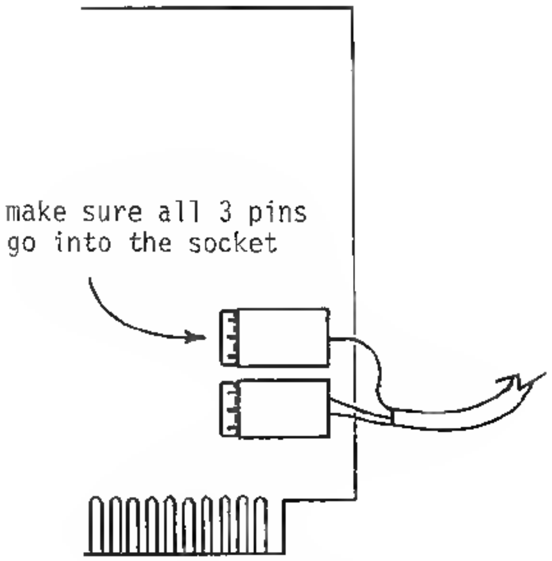
Please address all comments and suggestions to the appropriate person as indicated above.

# CONTENTS

This manual is published in a three-hole punched format, and is furnished with a reusable binder. Three-hole punched manuals have long been standard in the computer industry due to their versatility. They are easy to issue correction pages and addendums for, and the user can combine several manuals on similar topics into a single binder. ALF Products is proud to continue this fine tradition. We hope you will enjoy this format, and encourage other manufacturers to see the light.

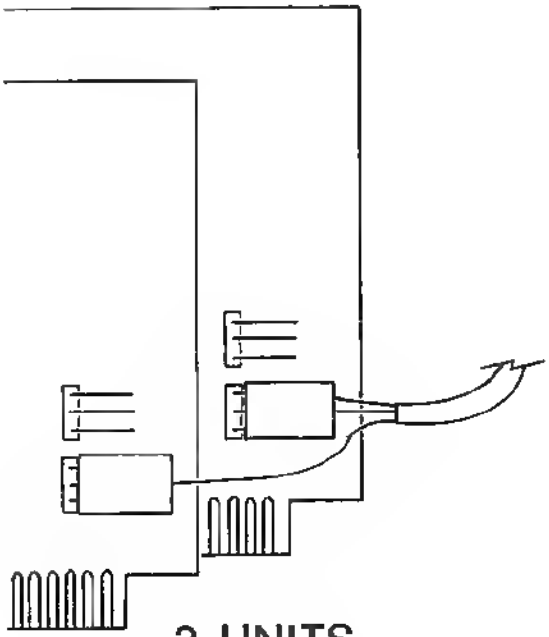
# 1

# INSTALLATION

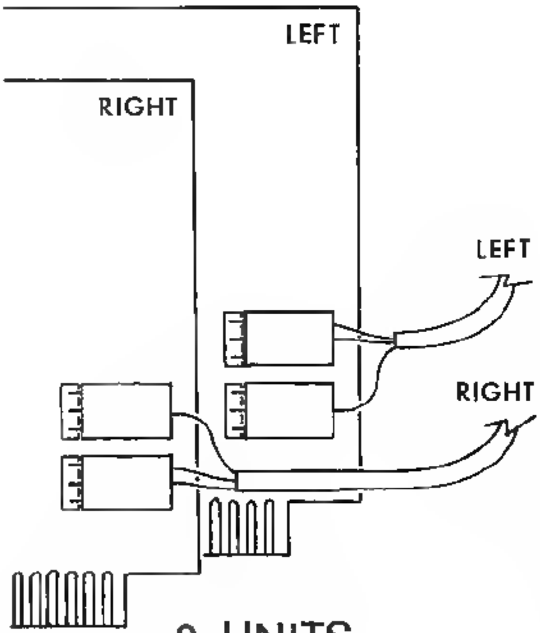


make sure all 3 pins go into the socket

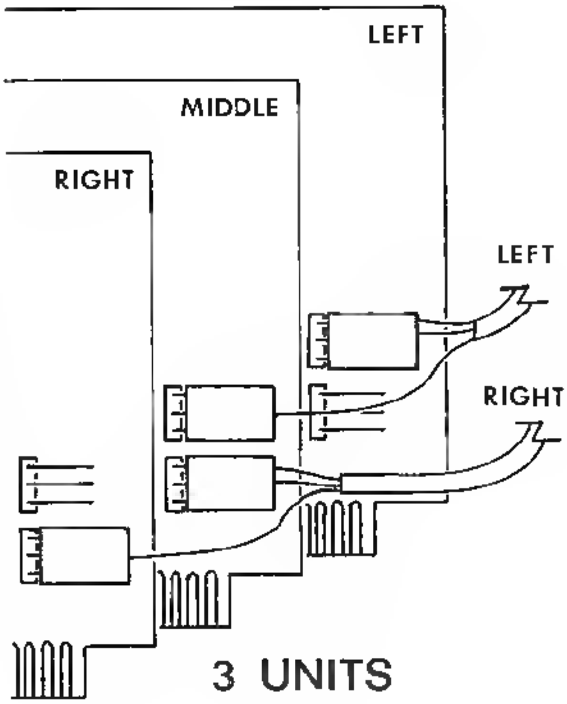
1 UNIT
MONO AND STEREO



2 UNITS
MONO



2 UNITS
STEREO



3 UNITS
STEREO

**THIS MANUAL DOES NOT COVER USE OF THE APPLE II COMPUTER. READ THE MANUALS SUPPLIED WITH YOUR APPLE, AND FAMILIARIZE YOURSELF WITH ITS USE, BEFORE CONTINUING.**

**PLEASE READ THIS ENTIRE SECTION BEFORE BEGINNING.**

Installation of your Apple II compatible music synthesizer is easy. Just follow these instructions:

1. You will need an audio amplifier and speakers or a home hi-fi system. One or two synthesizers can be used with a monophonic (mono) amplifier; and one, two, or three synthesizers can be used with a stereophonic (stereo) amplifier. Turn your amplifier off and the volume all the way down.

2. Turn the Apple off and remove the top cover.

3. Attach the audio output cable(s) to the synthesizer(s). One of the four drawings on the opposite page shows how the cables should be connected depending on how many synthesizers you have and what type of amplifier you're using. You'll notice that the connectors on the end of the audio cable can be plugged into the 3-prong connectors on the synthesizer circuit card in either of two ways: with the slots in the plastic housings toward the circuit card or away from it. You may plug them in either way. Just be sure all three prongs go into the three holes in the plastic connector.

4. Plug synthesizer(s) into expansion slot(s). Any slots may be used, but when using more than one synthesizer all slots used must be adjacent (see chart on next page). Route the cable(s) out through one of the holes in the back of the Apple. Replace the top cover of the Apple.

5. Plug audio cable(s) into amplifier or home hi-fi system. Any of a variety of inputs may used, such as Aux (or Auxiliary), Tuner, or Tape Play. Do not use Phono (phonograph) inputs. When two or more units are used in a stereo system, connect one cable to the Left input and the other to the Right input of the same type (e.g. Aux left and Aux right) as indicated on the opposing page. When using one unit in a stereo system, use either left or right input; and set the amplifier to "mono" if desired. When using the synthesizer, set the amplifier to select the input used (Aux or Tuner, or Tape for "tape play" or "tape in").

6. The synthesizer is supplied with several programs, on cassette tape or on disk. These programs are written to run using Integer BASIC. (Optionally,

programs are available for use with Firmware Applesoft. In this manual, these will be referred to as the Applesoft versions although they will not work with the version of Applesoft supplied on cassette tape for use with Integer BASIC Apples. Note that when using Applesoft, **FP** must be typed anywhere this manual says to type **INT**.) Each program which uses the synthesizer has a line which contains information regarding the slot number of your synthesizer, and some also have the number of units being used. This line is always located at line 1Ø. As supplied, all programs are for use with one synthesizer plugged into slot 4. If you are using more than one synthesizer, or if you have one synthesizer but it is not in slot 4, you will need to change some of the programs. Each program must be loaded, line IØ modified, and then saved. At the beginning of the instructions for each program in this manual the exact procedure required is explained. However, the variable SLOT (and sometimes UNITS) is used in each such procedure. To determine the value of SLOT and UNITS for your particular system, use the chart below.

| | | UNITS=I | UNITS=2 | UNITS=3 |
|---|---|---|---|---|
| SLOT=Ø | Synthesizers in slots: | Ø | Ø, 1 | Ø, 1, 2 |
| SLOT=1 | Synthesizers in slots: | 1 | I, 2 | 1, 2, 3 |
| SLOT=2 | Synthesizers in slots: | 2 | 2, 3 | 2, 3, 4 |
| SLOT=3 | Synthesizers in slots: | 3 | 3, 4 | 3, 4, 5 |
| SLOT=4 | Synthesizers in slots: | 4 | 4, 5 | 4, 5, 6 |
| SLOT=5 | Synthesizers in slots: | 5 | 5, 6 | 5, 6, 7 |
| SLOT=6 | Synthesizers in slots: | 6 | 6, 7 | |
| SLOT=7 | Synthesizer in slot: | 7 | | |

**IMPORTANT:** When changing line 1Ø you must load the program, change line IØ carefully making sure the length of the line is not changed, and then save the program. You must not save a program after it has been run, since it has then modified itself and thus will not contain many important statements which were originally present.

7. Turn your amplifier on. You are now ready to use the INTRODUCTION program. The INTRODUCTION section (which follows this section) contains instructions on running INTRODUCTION.

## OPERATING TIPS

Plug your Apple and amplifier into the same electrical outlet if possible. Differences in ground potentials can cause difficulties when different outlets are used. If different outlets must be used, or if the amplifier does not have a three-prong (grounded) power cord, do this: when removing the synthesizer from the Apple, always unplug the audio cable from the amplifier first. Similarly, plug the synthesizer into the Apple prior to connecting the audio cable into the amplifier.

**Always turn the Apple off before inserting or removing any circuit card.**

Some of the parts used on the synthesizer are static sensitive. Protection against normal static levels is provided by other components on the circuit card. No part should be removed from the unit except the audio cable. Otherwise, damage could result unless special anti-static precautions are carefully followed.

**Any Apple circuit board can be damaged by excessive static.** This particular circuit board has been carefully designed to minimize the possibility of damage (since only LS TTL type inputs are connected to the edge connector). However, walking across a carpet while holding an Apple circuit card can "charge" you and the card to voltages high enough to damage any electronic circuit. Therefore, you should always hold the circuit card in one hand, and touch the metal case of the Apple power supply with the other hand prior to inserting a board in the Apple. This will allow the static charge to be drained through the third prong (ground prong) of the power cord, rather than through the circuit card and the Apple circuits.

Avoid dropping the synthesizer onto a hard surface or severely jolting the unit. Otherwise the crystal may be damaged.

Should your synthesizer ever need repair, return the entire unit (including the audio cable and software) to your dealer or to ALF. Your dealer can repair the synthesizer if he is an ALF-authorized service agent; otherwise he can return it to our factory service department for prompt attention. Replacement parts, such as a new audio cable or owner's manual, can be obtained from your dealer or from the factory.

## PROBLEM CHECKLIST

1. Load the program you are using. List line 1Ø. Is is correct? If not, refer to the instructions for the particular program being used.

2. If no sound is produced, check the audio cable connections. If one of the three conductor plastic connectors has only two pins going into it (and the remaining pin or prong is unconnected) there will be no output. If this is the case, unplug the connector and plug it in correctly.

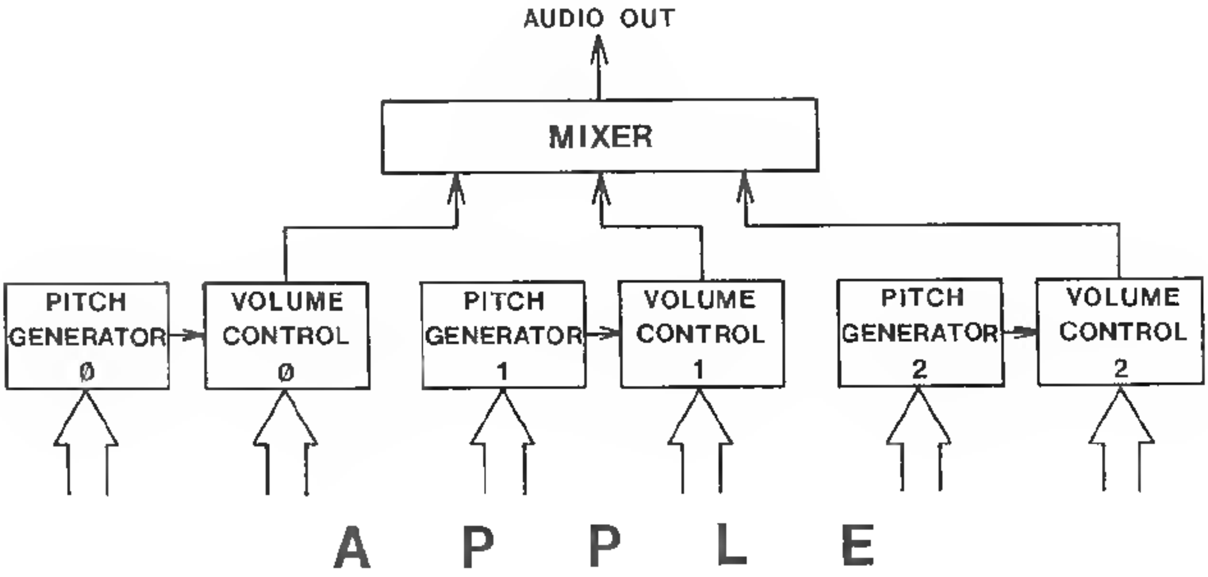
3. Check connections to the amplifier and all switch settings on the amplifier. Do the amplifier and speakers work with other sound sources? If not, replace.

# 2
# INTRODUCTION

The Apple Music Synthesizer is a simple three-channel synthesizer with direct hardware control of pitch and volume. Other effects can be produced with software. In normal use, each of the three channels is an identical and independent "monophonic synthesizer". A monophonic synthesizer is a musical instrument which can produce only one tone at a time ("mono"--one, and "phonic"--sound). Many conventional instruments are also monophonic. For example, trumpets, flutes, and clarinets can each only play one pitch at a time. In contrast, a piano can play several pitches at a time--unless you only use one finger. A piano is called a polyphonic instrument (from "poly"--many). The Apple Music Synthesizer is a polyphonic synthesizer since it can play three pitches at once, or up to nine simultaneous pitches using three synthesizers.

In order to create a synthesizer which is low cost, hardware control has been limited to control of pitch and volume. No other parameters can be controlled. Using software, pitch control can be used to create vibrato, sliding, or similar effects; and volume control can be used to create such effects as envelopes or tremelo. Since these are software-generated, in many applications it may be necessary to select only the most desirable effects to implement. The Apple may not be fast enough to perform the necessary calculations for all these effects, plus interpret a stored musical score, simultaneously. Note that waveform control is limited to square waves. (Pulse waves may be created in certain applications, see the CHROMA and BARE HANDED programming sections.)

A block diagram of the synthesizer is shown below:



## THE INTRODUCTION PROGRAM

A program named INTRODUCTION is supplied with the synthesizer. This program will introduce you to various technical terms used in music synthesis. Each term is explained and demonstrated with the synthesizer.

To run this program, you must have 24K or more memory. If you are using a DISK II, you need 36K or more. (Using the Applesoft version, these figures are 2ØK and 32K.)

First, load the program from disk or cassette tape. List line 1Ø. It will be 1Ø SLOT=4. Find the proper SLOT value for your system using the table in the INSTALLATION section. Carefully retype the line changing only the digit 4 to the proper digit for your system. Now save the program on your disk. If you do not have a DISK II, save the program using your own recorder to improve loadability. The program is now configured for your system, and can be run any time you like without having to change line 1Ø. If you ever change the slot position of your synthesizer(s), or purchase an additional synthesizer, you should do this configuration procedure again. (A note for perfectionists with three synthesizers: use a slot value one higher than normal to place the sound in the "middle".)

All instructions needed to run the INTRODUCTION program, once it has been properly configured as described above, will be displayed on the screen when the program is run.

## OTHER PROGRAMS

**ENTRY** is the most advanced program supplied with the synthesizer. It is used to enter songs (usually from sheet music) and play them. Entered songs can be saved on (and loaded from) cassette tape or disk. Full editing features are available.

**PLAY** is used to play songs entered with ENTRY. Although ENTRY can also be used to play songs, PLAY has the advantage of being significantly shorter than ENTRY. Thus, it is faster to load and it allows songs entered on systems with more memory to be played even if they cannot be loaded with ENTRY. PLAY has no editing features, but it has a more general "play" command which, when used in conjunction with DISCO, allows songs to be played in sequence.

**DISCO** creates a text file (execute file) which, in conjunction with PLAY, allows songs to be played in a specified sequence. It can also randomize the sequence. When used with a Timing Mode Input Board or similar Timing Mode arrangement, whole "albums" of songs can be played back using a single command.

**PERFORM** is used from BASIC programs to play songs. Songs created with ENTRY (or by any other means) can be played back using a CALL within your own BASIC program. It can also be used to create complex multi-channel sound effects.

**CHROMA** is used from BASIC programs to create complex sounds. Effects not possible with ENTRY, PLAY, or PERFORM can be created using CHROMA, processor speed allowing. Although far more complex to use than any of the other programs, CHROMA allows access to virtually all functions available on the synthesizer.

Complete programming specifications for the synthesizer are presented in the **BARE HANDED** section. Those who wish to program the synthesizer "bare handed" (that is, without any ALF-supplied programs) will find the hardware programming specifications they need to write their own assembly language or BASIC programs in this section.

# 3

# ENTRY

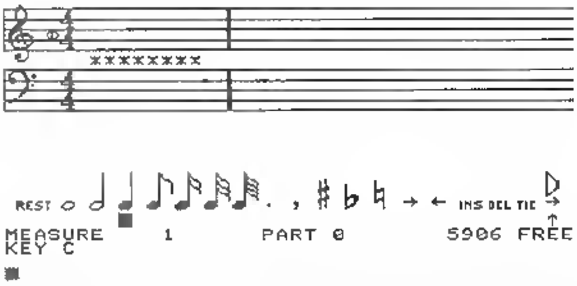
The ENTRY program is used to enter and play songs. Notes, rests, and other musical parameters are entered in a convenient sheet-music type format displayed on the screen (video monitor), and selected from a "menu" of available notes which is also shown on the screen. Songs entered can be stored on (and loaded from) cassette tape or disk. A variety of other functions are available for editing, stereo selection, and so forth.

To run this program, you must have 24K or more memory. If you are using a DISK II, you need 32K or more. (Using the Applesoft version, these figures are 32K and 4ØK.) Very detailed graphics are presented on the screen, so it is recommended that a black and white monitor (such as the Sanyo VM42Ø9 or VM4215) be used rather than a television set, although good results have been obtained using the Sup'r'mod II UHF channel 33 TV interface unit (from M&R Enterprises) and the Sony Trinitron model KV 1513 color television.

First, load the program from disk or cassette tape. List line 1Ø. It will be 1Ø SLOT=4 : UNITS=1. Find the proper SLOT and UNITS values for your system using the table in the INSTALLATION section. Carefully retype the line changing only the digits 4 and 1 to the proper digits for your system. (If you have a Timing Mode Input Board, list line 2Ø. It will be 2Ø TSLOT=8. Carefully retype the line changing only the digit 8 to the slot number of your Input Board.) Now save the program on your disk. If you do not have a DISK II, save the program using your own recorder to improve loadability. The program is now configured for your system, and can be run any time you like without having to change line 1Ø (or 2Ø). If you ever change the slot position of your synthesizer(s) (or Input Board), or purchase an additional synthesizer or an Input Board, you should do this configuration procedure again.

## ENTERING A SIMPLE SONG

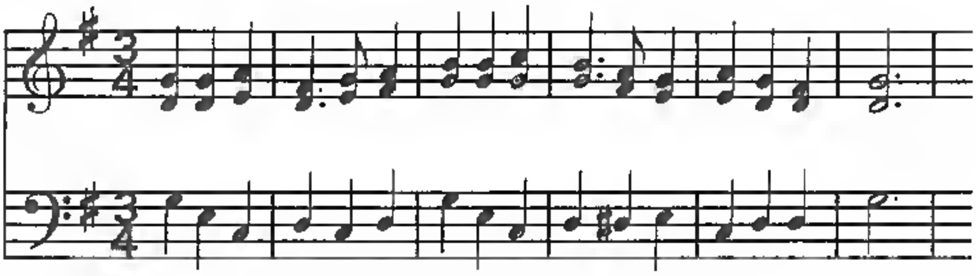
Load the program if it is not currently in memory. Type RUN and press return. The screen will go to hi-res graphics mode and display:



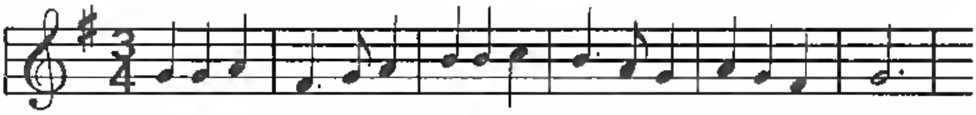
```
MEASURE    1      PART 0        5906 FREE
KEY C
```

The number in front of "FREE" will vary according to memory size and other factors. It indicates the number of notes which can be added, and will be constantly updated as you enter and edit the song.

The first six measures of "America" are shown below:



In order to enter the piece using ENTRY, it is first necessary to break the piece up into "parts". Each part is an independent melodic line in which at most one note is played at a time. It is best to choose each part so it is consistently from the same melodic line in the music. This allows you to select appropriate envelope settings for each line later on. The first part, called Part Ø, is shown below. It is the main melody.



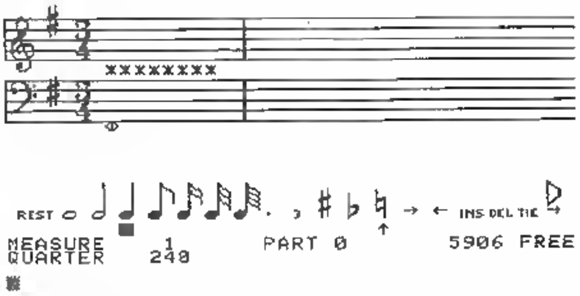
To begin entering a new song, type NEW and press return. ENTRY will display "NUMBER OF PARTS?". Just press return. This will make the song have only 1 part (part Ø). ENTRY now displays "SUGGESTED SPEED?". Since we don't really know what the playback speed should be yet, just press return. ENTRY will assume a speed of 255 (the slowest speed). ENTRY now displays "TITLE LINE 1". If you wish, you can type in a line which will be shown on the screen when the song plays. If you're not in the mood, just press return. The title lines can always be entered (or changed) later. ENTRY will then ask for title lines 2 through 4. Type titles if you like, or just press return for each line.

Part Ø can now be entered. Note that under "MEASURE 1" the screen shows "KEY C". If you turn paddle 1's knob, a small flying saucer will move up and down to the left of the two 4/4's. (If you get paddle Ø by accident, then a small arrow will move left and right instead. This doesn't matter. Try again with the other knob.) This flying saucer is called the "cursor", and it is very important. The cursor is a "pointer" to a particular item in the song. Currently, it is pointing to the KEY C before the 4/4. The key of C is a "neutral" key having no sharps or flats, and thus shows only as a blank space right before the 4/4.

Type KEY:1S and press return. A sharp sign will appear before the 4/4, and the cursor will move over to the 4/4. KEY:1S directs ENTRY to write a key signature of 1 sharp (S means "sharp", and F would be used for "flat"). This key signature is written over whatever item the cursor is on. Since it was on the KEY C, the

KEY C is overwritten with a KEY 1S.

When the KEY 1S is written, the cursor moves on to the next item in the song, which is a time signature of 4/4. The place on the screen which used to show KEY C now shows TIME 4/4 since the cursor is over the 4/4. "America" has a time signature of 3/4, so type TIME:3/4 and press return. The 4/4 will promptly change to 3/4, and the cursor will move on to the next item. The screen now looks like this:



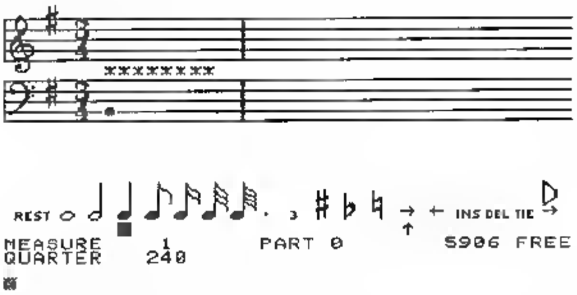
MEASURE 1 PART 0 5906 FREE
QUARTER 240

You've only been at this for a few seconds, and already you've told ENTRY two very important facts about "America", the song you're entering. Without these details it would be very difficult to enter the song properly. Why, you're probably half way to being a professional musician, if you weren't one when you started.

Now the cursor is at the first of eight asterisks (*) displayed between the treble and bass staffs, and the item the cursor is at is a QUARTER 24Ø. These eight items are special goodies that describe things about the song which don't display well in sheet music format. This particular one indicates how long a quarter note should play (24Ø time units per quarter note). While you will eventually want to learn about these, they are not important now, and it is best to skip over them at present. This is done using one of the paddles.

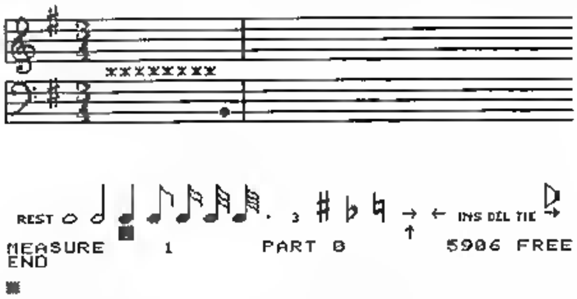
Turn one of the paddle knobs back and forth. If the arrow above "MEASURE 1 PART Ø 53Ø6 FREE" moves left and right, you're turning paddle Ø, the "menu paddle". If the flying saucer cursor moves up and down, you're turning paddle 1, the "note paddle". Place the menu paddle (paddle Ø) on your left and the note paddle (paddle 1) on your right. Usually you'll have your left hand on the menu paddle and your right hand on the note paddle; sometimes you'll have to let go of the paddles to type on the keyboard (probably not very often). Turning a paddle knob with one hand is almost always followed by pressing a paddle button with the same hand. You see, turning the knob selects something (a menu item when turning the menu paddle, or a note position when turning the note paddle), and then pressing the button tells ENTRY to look at the position of the knob and do whatever it is set for. Since the two paddles are used for different purposes, you always press the button of the knob you have just adjusted in order to activate the function you adjusted the knob to indicate.

For example, using your left hand only, position the menu paddle so that the upward pointing selection arrow points to the right pointing arrow in the menu. The screen will look like this:



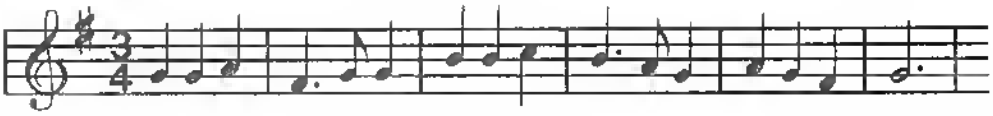
```
MEASURE      1          PART 0          5906 FREE
QUARTER    240
```

(The position of the flying saucer and the number of notes of FREE space available may be different than shown.) The right pointing arrow is used to move the cursor to the right. To cause a right movement, press the menu button using your left hand. The cursor will move right from an asterisk meaning QUARTER 240 to an asterisk meaning GAP 65535. To move the cursor right again, press the button with your left hand again. The cursor now moves to TRANSPOSE Ø. Press the button several times. The following items will appear: ATTACK 8192, DECAY 5Ø, VOLUME 55ØØØ, SUSTAIN Ø, and RELEASE 5Ø. (Most of these items specify an envelope. Envelopes are explained in the INTRODUCTION program.) Pressing the menu button again moves the cursor past the last of the 8 asterisks, and END appears under MEASURE 1 to indicate that the cursor is now at the end marker (that is, at the end of the song). This is where we will begin entering the notes of part Ø. The screen should now look like this:

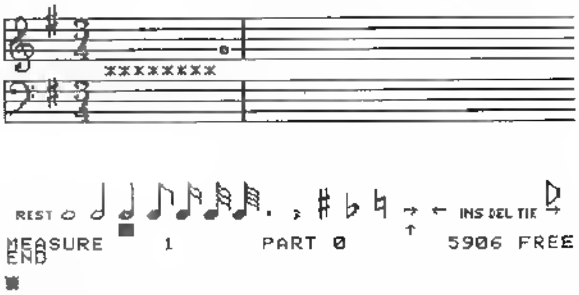


```
MEASURE      1          PART 0          5906 FREE
END
```

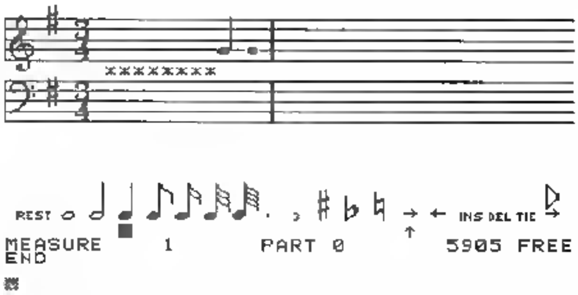
If it doesn't, you probably didn't start with RUN ENTRY like you should have. (The position of the flying saucer and the upward arrow, and the FREE number are not important.) Ready to really get into entering sheet music? Here's part Ø again, just as a reminder:



Using your right hand, turn the note paddle until the flying saucer is on the second line from the bottom of the treble staff, like this:



This is where the first note of part Ø should be. Still using your right hand, press the note button. A quarter note will appear at the second line, and the cursor will move over to the right. The pitch for that note is heard if you've got your synthesizer plugged in and your amplifier set up right. The screen now looks like this:

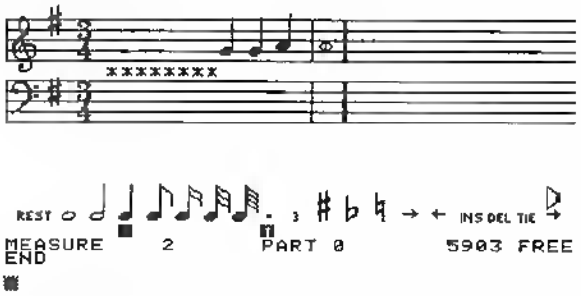


Normally when you type in something like TIME:3/4 or when you press the note button, the time signature (or note or whatever the cursor is pointing at) is written over and thus erased. However, erasing the end marker is not fun, so ENTRY automatically inserts the note (or whatever is entered) in front of the end marker. Then, when the cursor moves to the right, END is still shown under the MEASURE number since the end marker is still there.

It's time to give your left hand something to do for a while. Just for fun, position the arrow under the left pointing arrow in the menu (using the menu paddle, of course). Press the menu button. This will cause the cursor to move left. Under MEASURE 1, NOTE GN3 24Ø is displayed. That's the note you entered, a G Natural in the 3rd octave (the octave number is an ALF creation and has nothing to do with the rest of the world). "Natural" means it is neither sharp nor flat. The 24Ø indicates the number of time periods long the note should be during playback. (When you press the note button to enter a note, it is just played for as long as you hold down the button.) Remember the QUARTER 24Ø that said quarter notes should be 24Ø time periods long? Well, they obviously are. Move the menu arrow so it is under the move right arrow and press the menu button. You're back to the end marker now. Isn't this exciting?

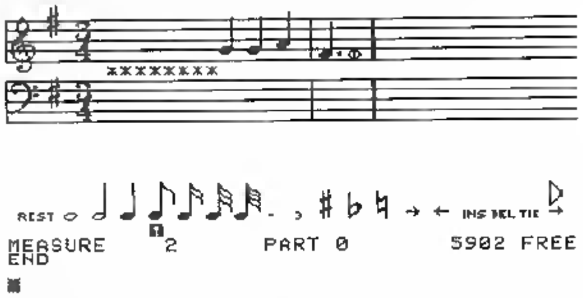
On to the second note. You've probably still got the note paddle set so the flying saucer is on the second treble line. (If not, move it until it is.) Press the note button. The next note is heard and appears on the screen. It is the same as the first note. Now, turn the note paddle until the saucer moves up one click to the space above the second line. Press the button to enter this note (are you doing all this note-paddle stuff with only your right hand?). Not only do you hear this note and see it on the screen, but also a bar appears between the note and the flying saucer. This is because TIME 3/4 means that there are 3 (3/) quarter notes (/4) in a measure. Since the measure is now full, ENTRY automatically shows a measure bar. You'll notice that there is a bar at this point in the sheet music, too. If ENTRY and the sheet music don't seem to agree on where to put the bars, then either the sheet music has a typo (that is, a wrong note) or you've skipped a note or made some other error. Just by watching the measure bars you can be confident that you haven't made any timing mistakes.

If you're looking ahead at the music for part Ø, then you know that the next note isn't a quarter note. It's a dotted quarter note, which plays for as long as a quarter note plus an eighth note. (A dot always means to add the time of the next shorter note to the note length shown.) You may well be wondering why ENTRY always makes a quarter note whenever you press down the note paddle button. Well, it's because a block is lit up under the quarter note in the menu. When you press the note button, a note as long as selected in the menu (shown by one or more blocks) is entered. To change from a quarter note to a dotted quarter note, you position the menu arrow under the dot, which is just to the left of the "3", and press the menu button (left hand, remember?). A block instantly appears under the dot, and the block under the quarter note remains. The screen now looks like this:



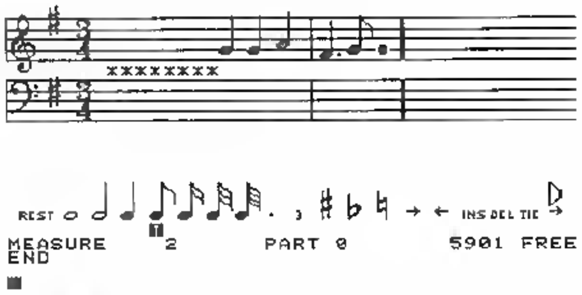
Okay, fire away. Move the note paddle down two clicks to the space under the second treble line, and press the note button. You see how you switch between the left and right hand, usually rotating a knob and pressing a button with the same hand? Since you generally keep your hands on the two knobs, you can enter notes really fast. You don't even have to look at the screen when you are entering several notes of the same length, because you can just count the "clicks" the Apple's built-in speaker makes at each line or space on the staff. (On some other music systems, you have to type in codes like the GN3 you saw on the screen a while back, and this requires that you memorize the octave numbers.)

To enter the next note, position the menu arrow to the eighth note and press the button (I'm not going to remind you to use your left hand, since you've probably got that all straight by now). The blocks under the quarter note and the "dot" go out, and one appears under the eighth note, like this:



```
MEASURE   2        PART Ø        5902 FREE
END
```

Move the note paddle up a click to the second line, and press the button to enter the eighth note. The screen now looks like this:

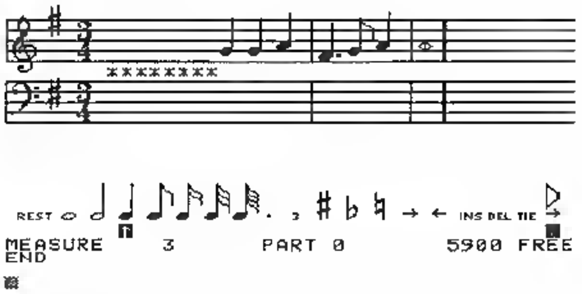


```
MEASURE   2        PART Ø        5901 FREE
END
```

Let's take a look back. Move the cursor left one. (You know how to do it, we just did it a while back to see the first note displayed as NOTE GN3 24Ø.) The eighth note shows up as NOTE GN3 12Ø. It's the same as the first note in this part except it's half as long (only 12Ø time periods). That dotted quarter note we're coming up to should be a quarter (24Ø) plus an eighth (12Ø) long. Back up again to see it. Yep, NOTE FS3 36Ø. But wait, doesn't FS3 mean an F sharp in the 3rd ALF octave? We didn't enter a sharp note. The reason for this is that the key signature indicates that all F's should be sharp. So, ENTRY automatically enters F's as being sharp, without the user having to specify it. Of course.

Back up three more times to get to the first note. Now, position the menu pointer to the rightmost menu item, a little speaker with a right arrow under it. Press the menu button, and a small block appears under the speaker/arrow. Curious? Position the arrow for right movement, and press the menu button five times to go past all the notes (do it fairly slowly, and pause a little extra at the dotted quarter note). You'll hear the first 5 notes of "America". The speaker/arrow activates playback during right movement. The timing of the notes is still dependent on how long you press a button down, but don't worry. It won't be during actual playback. You don't believe me, do you? All right, type PLAY and press return. ENTRY shows "SET SPEED (255) AND PRESS BUTTON". Crank the menu paddle up all the way (if may not actually get up to 255, but who cares?). ENTRY doesn't happen to mention which button you should press, but it is the menu button. Trust me. Punch it and ENTRY will play the song. A little slow, perhaps, but we'll know better next time.

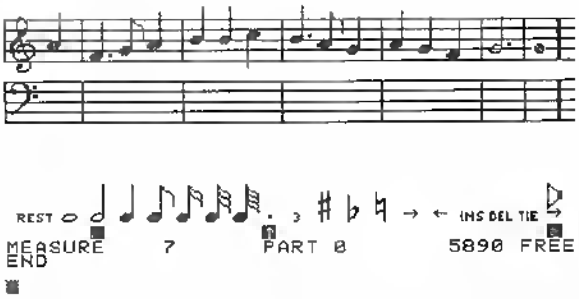
Let's put in another note. I'll bet you're thrilled at the prospect. Just select a quarter note using the menu paddle, flash the note paddle up to the space above the second treble line, and punch the note button. Here's a screen image just to make sure we're together:



MEASURE 3 PART 0 5900 FREE
END

Click up one to the third line. We're already set for quarter notes, so press the note button. Twice. Now, click up and press again (you should take a look at the music for part Ø again so you'll know what you're doing). That completes another measure. The display now shows MEASURE 4. This means the cursor is pointing to an item which is in the 4th measure. In this case, it is the end marker which is indeed in the 4th measure.

Faster now. Set for dotted quarter. Down a click and punch. Switch to eighth. Down a click and punch. Now quarter. Down a click, punch, up a click, punch, down, punch, down, punch. Last measure. Set for dotted half. (In case you haven't noticed, you can't set for "dot" and then "half" because "half" turns off "dot". Set "half" first, then "dot".) Okay. Up a click, and punch. We're out of music (just the first 6 measures, remember?). Are you getting fast at it yet? You will. It's easy. Let's see the screen now:

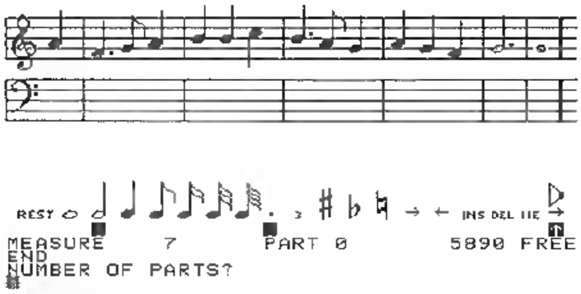


```
MEASURE      7          PART 0           5890 FREE
END
```

Type PLAY and press return. Let's try a speed of about 2ØØ now. Adjust the menu paddle to some number in the vicinity of 2ØØ. (Don't get too picky, it's not important to get exactly 2ØØ.) Punch the button, and the first 6 glorious measures issue forth.

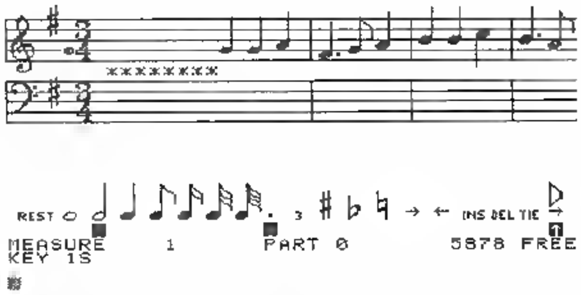
Rapture! Ecstasy! Sublime delight! (Where's my thesaurus?) Ah, the joys of music. And yet, that's just one part. Let's get on to THREE PARTS. Quick!

Fortunately, it is quick. First, we have to tell ENTRY that we want to add a second part. Type EDIT and press return. ENTRY responds by showing:



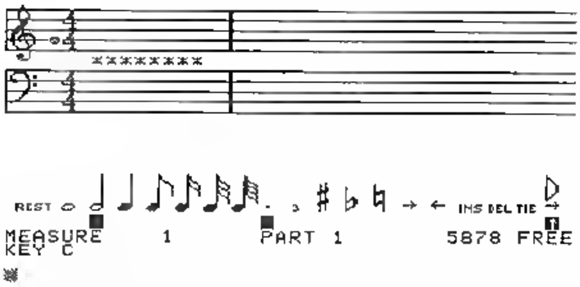
```
MEASURE      7          PART 0           5890 FREE
END
NUMBER OF PARTS?
```

Since we want 2 parts, type 2 and press return. ENTRY then asks for the "suggested speed". Just press return to leave this as it was before. It will then display each of the four title lines. Just press return each time. The screen now shows:



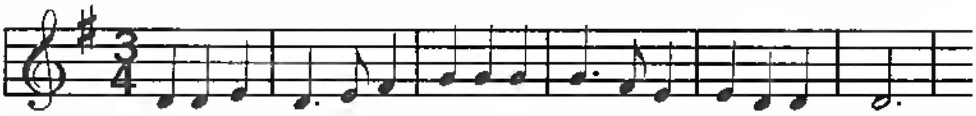
```
MEASURE      1          PART 0           5878 FREE
KEY 1S
```

This is the beginning of Part Ø, the part you just entered. The part just created is Part 1. To see Part 1, type PART:1 and press return. The screen shows:

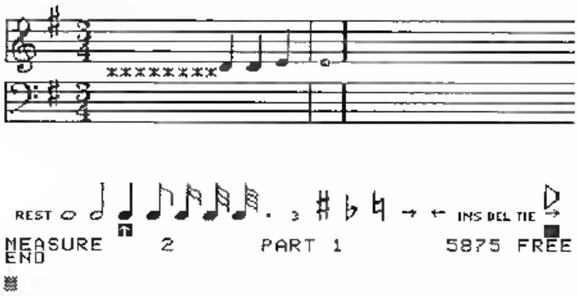


MEASURE 1 PART 1 5878 FREE
KEY C

This is just like Part Ø looked originally, except there are fewer notes of "free" memory, and the screen shows "PART 1" instead of "PART Ø". You now proceed in the same fashion as before. Type KEY:1S (return) and TIME:3/4 (return). The music for Part 1 is as shown below:

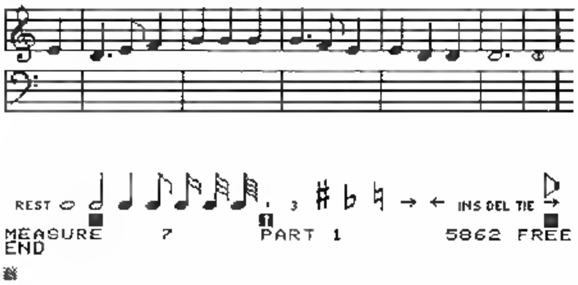


Use the right arrow function to skip over the eight asterisks, and enter the first three notes as usual. The screen should now look like this:



MEASURE 2 PART 1 5875 FREE
END

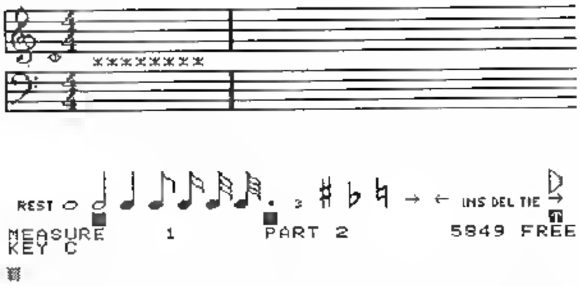
Type PLAY and press return. (As usual, set the speed and press the paddle button to start playback.) You'll notice that only the first measure is played. Playback always stops when the end of the highest numbered part is reached. Since we've only entered the first measure in Part 1, and Part 1 is the highest numbered part, only the first measure is played. Enter the remaining notes of this part in the usual fashion. The screen will look like this:



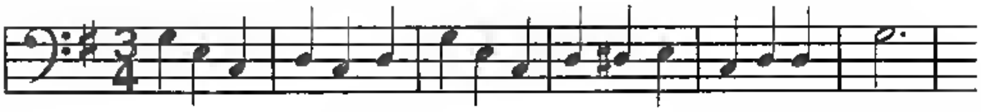
MEASURE 7 PART 1 5862 FREE
END

Type PLAY and press return. (I won't tell you to adjust the playback speed paddle since you've got that figured out already.) If there are any wrong notes, back up and correct them. (More details on correcting wrong notes will be given later in this section.) You're now ready to enter the third part.

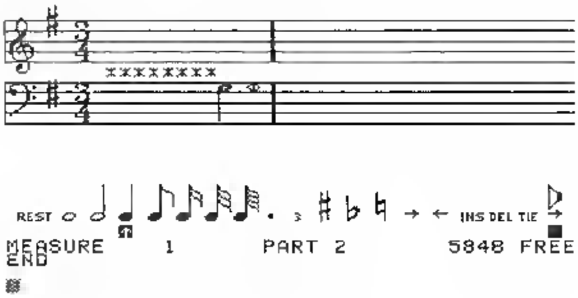
Type EDIT and press return. Ask for 3 parts this time, and then press return to skip the other questions. When Part Ø appears, type PART:2 to go to the third part. The screen shows:



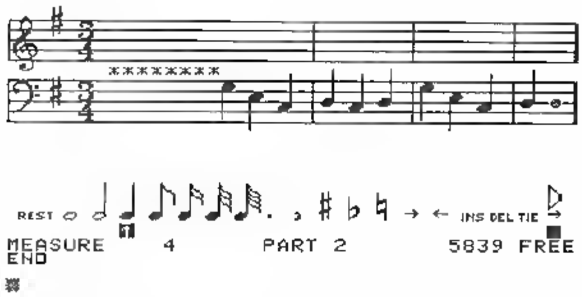
Begin as usual, typing KEY:1S and TIME:3/4, then skip the asterisks. Just for fun, type PLAY and press return. When you press the paddle button to begin playback, there is a brief flash and the hi-res graphics screen reappears. This is because the end of the highest numbered part (now Part 2) is reached immediately, since there are no notes entered in it yet. Now comes your big chance to use the "bass staff", which has been ignored up to this point. The bass staff is the lower five horizontal lines. The sheet music for Part 2 is shown below.



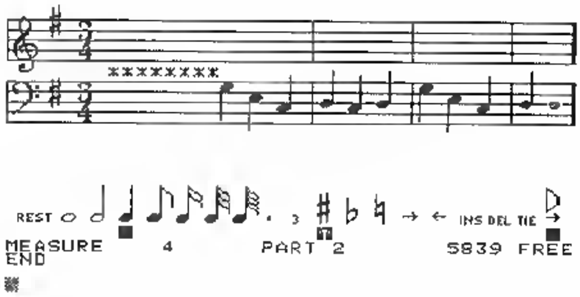
Enter the first note. The screen now shows:



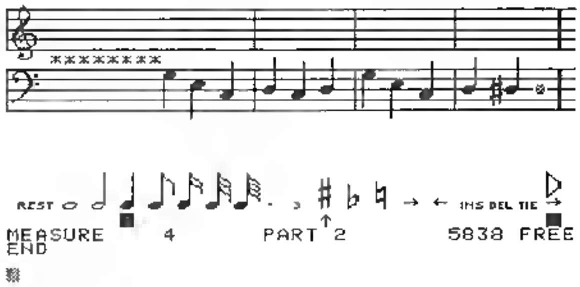
Enter the next nine notes. The screen shows:



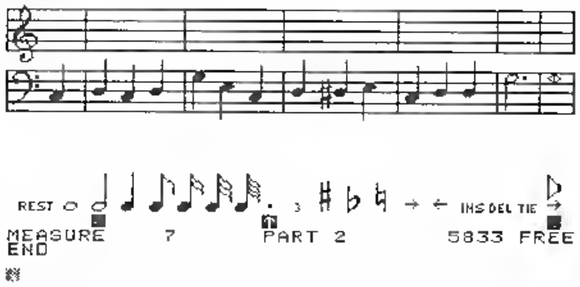
The next note is sharp, so use the menu paddle to light up the sharp sign in the menu, like this:



Now enter the note. The sharp sign in the menu disappears into hyperspace:



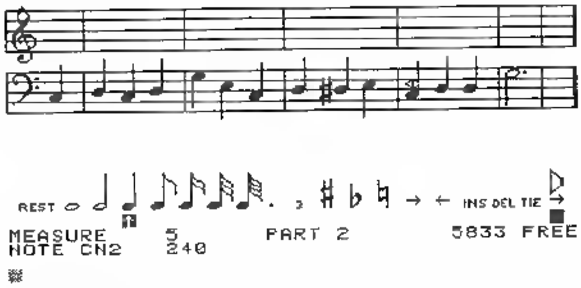
Enter the rest of the part. The screen shows:



Type PLAY to hear the song and check for errors.

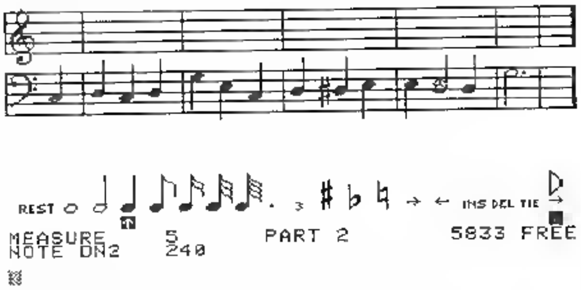
## CORRECTING MISTAKES

Back up to the first note in measure 5 (of Part 2). Let's say we want to change this note so it is at the next space up on the staff. First, set the menu notes for a quarter note, and put the cursor in the space above the note:



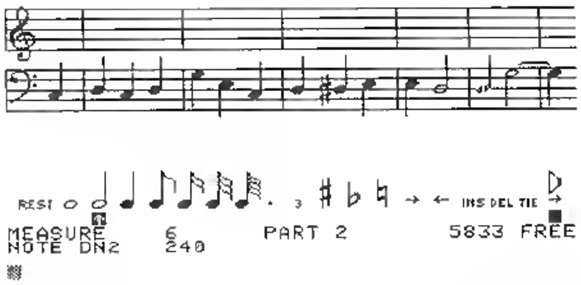
MEASURE 5 PART 2 5833 FREE
NOTE CN2 240

Now just press the note entry paddle button (paddle 1, of course). The old note is written over by the new note:



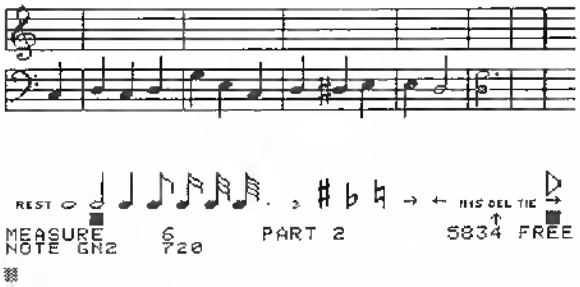
MEASURE 5 PART 2 5833 FREE
NOTE DN2 240

The rest of the song is not affected. Now, let's say we want to change the next note in the measure into a half note of the same pitch. Set for half note, position the cursor so it is over the quarter note's head (in order to get the same pitch), and press the button:



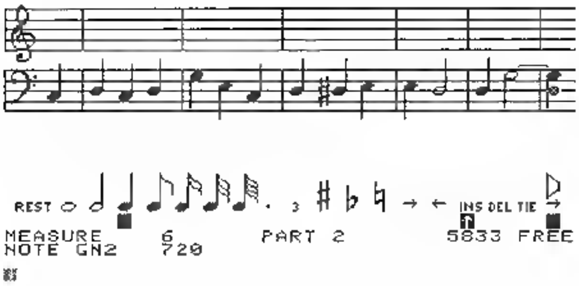
MEASURE 6 PART 2 5833 FREE
NOTE DN2 240

What if we want to get rid of the first note in measure 6 (where the cursor is now)? Just position the arrow for "DEL" and press the menu paddle button:



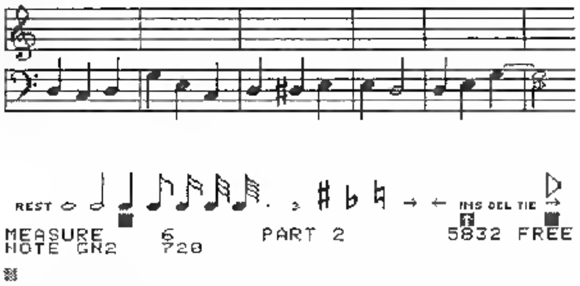
```
MEASURE   6        PART 2        5834 FREE
NOTE GN2  720
```

Now, let's change our mind and put it back. It was a quarter note, so set for quarter. Position the cursor on the middle bass staff line to get the same pitch. We need to insert the note, so put the menu arrow under "INS" and press the menu button to light up a block under it. Now just press the note button to enter a note as usual. Instead of replacing the note the cursor is at, the entered note will be inserted in front of it because "insert" mode is on:



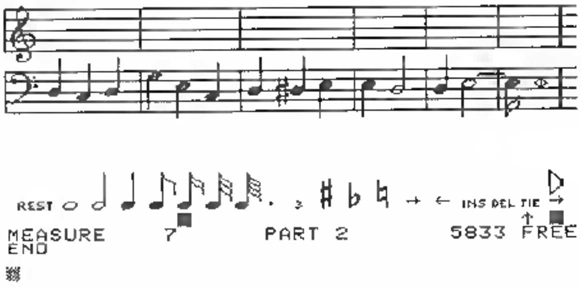
```
MEASURE   6        PART 2        5833 FREE
NOTE GN2  720
```

Click the note paddle up one, and press the note button again. Another note is thus inserted:

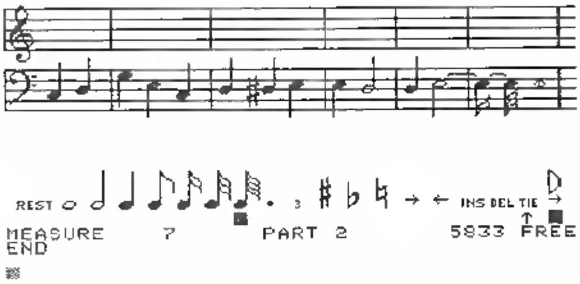


```
MEASURE   6        PART 2        5832 FREE
NOTE GN2  720
```

Now press the menu button while the arrow is pointing at "INS". The block of light goes off. Enter a note. Since "insert" mode is no longer on, the old note is replaced by the new one. Next, back up one and delete the last one of the two similar quarter notes so the next demonstration will be more clear. Let's change the remaining quarter note to a half note. We could set for half note and reenter a half note over the old quarter note, or. . . leave the menu setting at quarter note, aim the menu arrow at "TIE", and press the menu button. There is a beep, and the cursor backs up. Now press the menu button once more to do "TIE" again. The current setting (quarter note) is added to the note the cursor is at. Since it was originally a quarter note and we added a quarter note, it becomes a half note. (Note: the first time you pressed the button for "TIE", the cursor was not at a note or a rest, so the tie could not be done. Since you usually tie the last entered note, ENTRY backs up one when you do an illegal tie, allowing you to just press the button twice to tie the last note.) Now set the menu for a sixteenth note. Aim at "TIE" and press the button twice. The note is now a half note tied to a sixteenth:



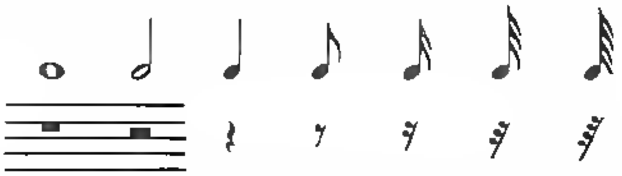
The vertical position of the note paddle cursor is not important during a "tie" since the note paddle is not used. It is important to note that although the half note tied to a sixteenth note is shown as "two" notes, it is really only one. If you back up and look at it, you will see that the length shown is 54Ø time periods, which is a half (48Ø) plus a sixteenth (6Ø). In fact, the little curved line between notes always means that the multiple notes shown are really only one note. This happens on tied notes and on notes that have part of their duration in one measure and the remainder of their duration in the next measure. Tie in a sixty-fourth to the last note, and you'll see that more than two "notes" can be tied together to display a single note:



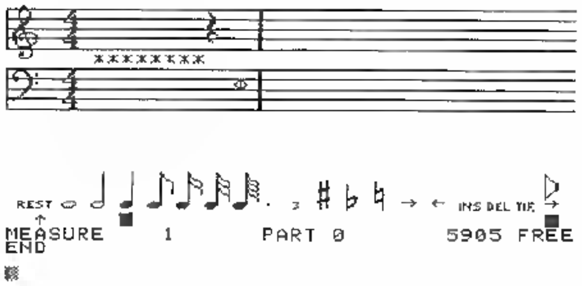
In general, mistakes are corrected (or any desired changes are made) by using the above functions (change a note, insert a note, delete a note, and tie additional duration to a note) until the screen shows what you want. When using these functions, only the current part is affected. In fact, the only functions available in ENTRY that affect anything besides the current part are the NEW, EDIT, STEREO, and SPEED commands which by their very nature must relate to the entire song.

## ENTERING RESTS

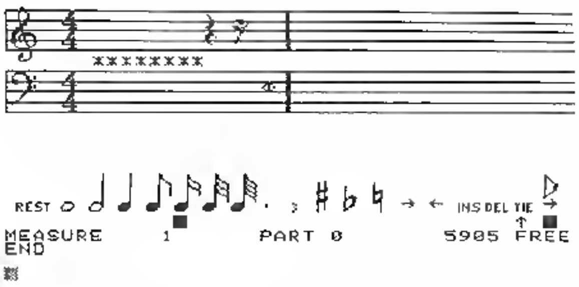
On occasion a part must sit around for a while and not play anything. This is called a "rest". Rests are entered in much the same fashion as notes. There are two main differences: the vertical position of the note cursor doesn't matter (since rests don't have any "pitch"), and the menu paddle is used to enter a rest, rather than the note paddle. Obviously, you point the menu arrow to "REST" and press the menu button to enter a rest. The duration of the rest is determined by the menu, just as the duration of a note is. Rests are displayed with different symbols than notes. They correspond like this:



Let's start on a new song. (Actually, "song" refers to a musical composition with lyrics. Technically, one shouldn't use "song" to refer to just any melody, but there isn't any simple word available. Musicians use "piece" or "work", apparently in an effort to avoid any disclosure that music is involved. In fact, all artists use "piece" and "work" to describe their creations.) Type NEW and press return. Press return 6 more times to avoid answering the useless questions. Skip over the key and time signatures, and the eight asterisks. Select quarter note, and press a REST. A quarter rest appears on the screen.



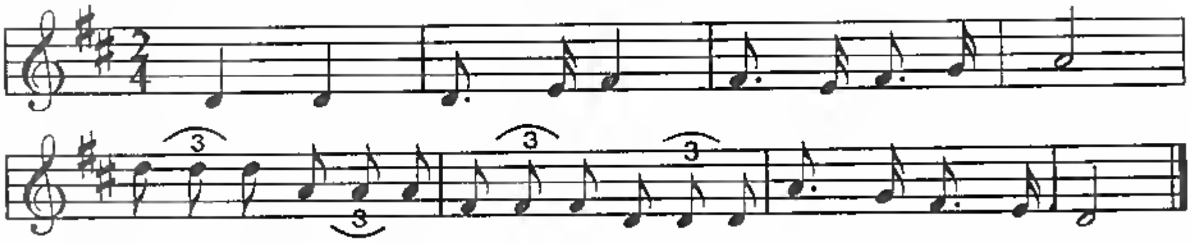
Now select sixteenth note duration and tie it onto the quarter rest. Oddly enough, the screen shows:



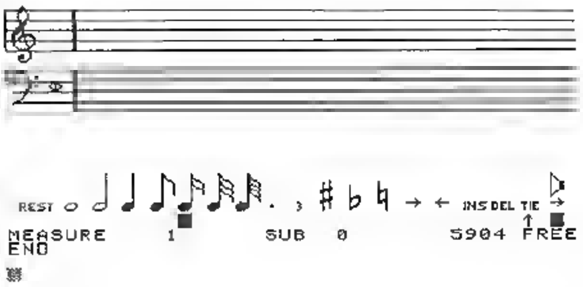
In traditional music notation, rests are never shown as being tied. This is because there is no difference between, for example, a half rest and two quarter rests during performance. The ENTRY screen display makes no distinction between a rest which is as long as a quarter plus a sixteenth, and two rests the first of which is a quarter and the second of which is a sixteenth. However, it takes only one "right movement" to skip a single tied rest, and two to skip past two individual rests. (Plus, two rests would take twice as much memory as a single rest.) Incidently, when a large number of rests are tied together (for example, in a part which doesn't begin playing until far into the song) the cursor will be at the last of the rests displayed, and the measure number will reflect the measure number the rest starts in. (This is true of notes, too.)

## SUBROUTINES

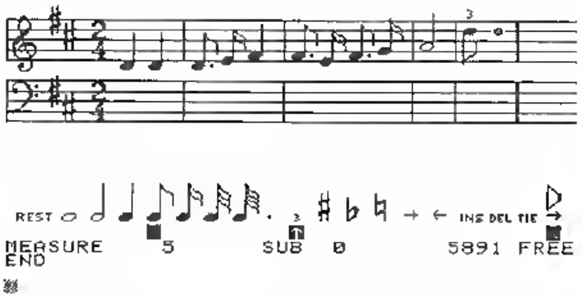
Most people are familiar with the song "Row, Row, Row your Boat". If you're not, become so. This song plays the same theme several times, and from several parts. It seems that one would have to enter this theme several times. Since repeated sections such as this are common in music, ENTRY has special provisions for entering them. The sheet music for this song is thus:



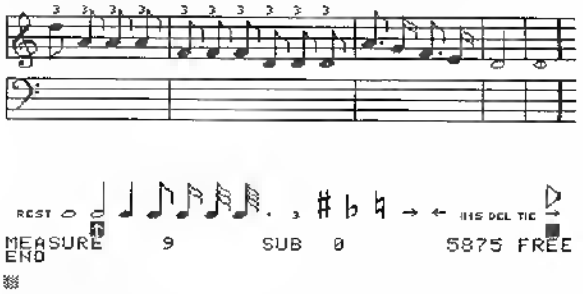
This theme must be entered in a special fashion which allows it to be played many times. This is done using a subroutine. Type NEW and press return several times (as usual) to start fresh. Now type SUBROUTINE:Ø and press return. The screen will show:



Type KEY:2S and TIME:2/4 to enter the key and time signatures. (Otherwise KEY:C and TIME:4/4 are assumed.) Enter the first four measures of the theme in the usual fashion. You'll notice that the next note is a triplet. Triplets are entered in the same fashion as dotted notes. Just light up the block under the "3" after selecting eighth note. Now press the note paddle button to enter the note. The screen will show:



The little 3 above the note indicates that it is a triplet. Conventional sheet music notation shows triplets with a curved arc above the three notes and a single 3. ENTRY puts a little 3 above each note. This is because ENTRY, unlike conventional notation, allows the presence of a single triplet note (that is, a single note with a duration equal to one of the notes of a conventional triplet set). Press the note button twice more to enter the remaining two triplet notes of that pitch, then enter the remaining three sets of triplets, and the rest of the theme. The screen will show:



Now type PART:Ø and press return to go to Part Ø. Type KEY:2S and TIME:2/4 as usual, and skip the 8 asterisks. Now type CALL:Ø and press return. A 9th asterisk appears. During playback, this CALL causes the theme entered into its associated subroutine to be played. (CALL:1 would play the theme entered into SUBROUTINE:1.) Type PLAY and press return. The basic theme is played. Now, type in another CALL:Ø after the first one. Type PLAY again and note that the basic theme is played twice.

Now EDIT the song to 2 parts. Type PART:1, KEY:2S, and TIME:2/4. This time, instead of skipping the 8 asterisks, step forward until TRANSPOSE Ø is shown. If we played the basic theme exactly the same in both parts, they would be hard to tell apart. So, type TRANSPOSE:24 and press return. The TRANSPOSE Ø is of course thus changed to TRANSPOSE 24. The transpose function raises all following pitches by the specified amount of quarter steps. There are 24 quarter steps per octave (2 quarter steps is the difference between two adjacent keys on a piano, including both black and white keys), so TRANSPOSE:24 will cause this part to be played one octave higher in pitch than the other part. Skip over the remaining asterisks. Part 1 is supposed to begin after Part Ø has already been playing for two measures. Select a whole note duration and enter a rest. It will show as two half rests due to the 2/4 time signature. Now type in two CALL:Ø's. Type PLAY. A two-part round will be played.

Let's add a third part. EDIT the song to 3 parts. Type PART:2, KEY:2S, and TIME:2/4. Skip to the TRANSPOSE setting again. Let's shift this part down one octave. Oddly enough, to transpose down you take the number of quarter steps you wish to transpose down, and subtract that number from 256. 256-24 is 232, so type TRANSPOSE:232. Now skip past the other asterisks. Punch in a whole rest, then press TIE twice to make it two whole rests (which will display as four half rests, again due to the time signature). Type in the usual two CALL:Ø's. Now just type PLAY to hear the full three-part round.

Perhaps you've noticed that you really didn't need the KEY:2S's in the three parts, since there aren't any notes anyway. You could have simply deleted the key signature if you prefer. However, often there are notes in the part, and in that case the key signature would be needed. In this particular instance, even the time signature could have been deleted without affecting the song. Naturally, the KEY:2S was needed within the subroutine, else the notes of the song would be incorrect.

Here are a few things you should know about subroutines. You can have 1ØØ subroutines numbered Ø through 99. Always begin with subroutine Ø and proceed by 1's. If you press RESET, or if you save a song and load it again, all the subroutine numbers will be readjusted so they do begin with Ø and proceed by 1's. A subroutine is created when the first SUBROUTINE command using its number is entered. All subsequent SUBROUTINE commands with that number merely cause the subroutine to be displayed and to be available for editing. (That is, the first SUBROUTINE command for any given subroutine is like the EDIT command for new parts. All future SUBROUTINE commands are like the PART command for parts.) Once created, a subroutine cannot be destroyed. The most you can do is delete everything in it. A CALL can be entered only to an existing subroutine. (That is, you can't even enter a CALL to a subroutine you haven't created yet.) Subroutines are not limited to notes and rests. You can put a TRANSPOSE function in a subroutine, for example. Some things, like key and time signatures, can be put in a subroutine to affect the notes entered in the subroutine, but they do not affect the notes entered outside the subroutine, even after a CALL to the subroutine. The summary of commands in this section tells the effects of each command.

Subroutines can be used in a much more complex fashion than shown in this simple example. For example, subroutines can contain CALLs to other subroutines. If a subroutine contains a CALL to itself, the song will repeat forever (unless the highest numbered part does not use a subroutine which CALLs itself, in which case the song will stop whenever the highest numbered part stops). NOTE: be sure there is at least one note or rest in a subroutine that CALLs itself; otherwise the playback routines will not continue processing all parts.

## LOADING AND SAVING SONGS

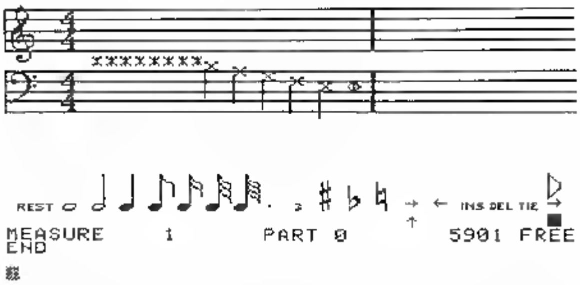
If you want to save Row, Row, Row then you should type SAVE and press return, if you want to save it on cassette tape. When saving a song to disk, it is necessary to specify a name. For example, you could type SAVE:ROW and press return. Names can contain any characters except comma, and can be up to 28 characters long. (Control letters and trailing spaces are ignored.) Disk specifications like ",D2" or ",S3,D2" can be added after the name if needed. Note that songs will appear in the catalog as Integer BASIC programs (even if your system doesn't have Integer BASIC) and will have names that begin with "M:". Songs are loaded the same way, using LOAD instead of SAVE.

The synthesizer is supplied with a few sample songs which can be loaded and played. Additional songs are available at extra cost.

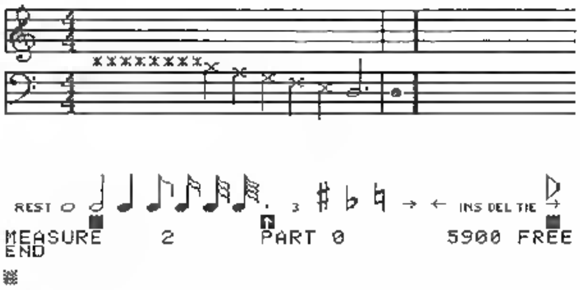
## ADJUSTING THE TEMPO

Let's say we want to enter the "row" theme to play twice as fast with the same paddle setting. That means each note will have to play for half as many time periods. Type NEW and press return as required, enter the key and time signatures, and you'll be at the QUARTER 24Ø function. Type QUARTER:12Ø. This will make all quarter notes be entered as 12Ø time periods instead of 24Ø (and thus take half the time, so the song will play twice as fast). The other menu notes' duration values will change proportionately. Skip over the other asterisks and enter the theme. Now type PLAY and use the same paddle setting as you did previously. The song does indeed play twice as fast. Type PART:Ø to get back to the beginning of the part, and skip over to the QUARTER function. Change it back to QUARTER:24Ø. You'll notice that all previously entered notes show as notes half as long as originally entered. Examine any note by moving the cursor to it. Notice that the length in time periods is still the same. You didn't change any of the notes, only the QUARTER function, so of course none of the notes have been altered. Obviously ENTRY stores notes based on their "time period" length, and just computes the proper note to display based on the QUARTER setting. (And the QUARTER setting determines the "time period" length of notes when they are entered.) Since none of the notes have been changed, the song will still play as it did before. In fact, you can skip right a measure or two (you might want to look up the MEASURE command in the summary of commands) and <u>insert</u> a QUARTER:12Ø. Notes before the QUARTER function will be shown as half as long as originally entered due to the QUARTER:24Ø, and notes after the QUARTER:12Ø will be shown as entered. None of this affects playback, but any <u>new</u> notes you might enter would be based on the current QUARTER setting. Remove the inserted QUARTER, if you put one there, and change the QUARTER at the beginning to QUARTER:12Ø as it was when the notes were orignially entered. Now type SPEED:2 and press return. This will multiply the "time period" lengths of all notes in all parts and subroutines by 2. Rest durations and QUARTER settings are also multiplied by the specified amount. Now the song plays twice as slow (also known as half as fast). In fact, it should look just like the original QUARTER:24Ø version, except that it used a subroutine and multiple parts. (CAUTION: the SPEED command can be tricky to use. See the complete description in the summary of commands.)

By typing in a QUARTER function wherever you need a different tempo, you can make the song play at different speeds from section to section. Just remember that the QUARTER function affects only notes which haven't been entered yet. Another way to get unusual note durations is by using the LENGTH command. Let's say you want to play five notes in the space of a single quarter note. A standard quarter note is 24Ø time periods long, so each of your five notes will have to be 24Ø/5 or 48. Unfortunately, there aren't any menu notes that are 48 time periods long. So, type LENGTH:48. The block(s) under the menu notes disappear to indicate a non-standard note length. All notes (and rests) you enter now will be 48 time periods long. Give it a try by making a new song and punching in five notes. The screen should look like this:



Since there is no representation for a note 48 time periods long, each note has a small X. To cease entering non-standard notes or rests, just activate any menu note. For example, put the menu arrow under the half note and press the menu button, then do the same for the dot (".") to select a dotted half note. Punch in a note, and the screen shows:



The measure bar shows that a full 4 quarter notes worth of duration have occured, verifying that the five funny notes took up one quarter note of time.

## ENVELOPES

Envelopes are a little complicated, and to really get the most out of your synthesizer is going to require a little study, some effort, a fair amount of calculation, and an awful lot of experimenting. Let's start at a very simple level. If you aren't completely familiar with standard synthesizer envelopes, run the INTRODUCTION program (see the INTRODUCTION section). Now that you're familiar with the terminology, here's how it applies to the various envelope commands. They are ATTACK, DECAY, SUSTAIN, RELEASE, VOLUME, and GAP. The first point of possible confusion is with the VOLUME function. It does not set the output volume like a volume control would. It sets the maximum loudness level reached during the attack stage (that is, the point at which the switch from the attack stage to the decay stage occurs). Both VOLUME and SUSTAIN specify a loudness level. SUSTAIN:Ø selects a very low level (soft), and SUSTAIN:65535 selects a very high level (loud). ATTACK, DECAY, and RELEASE specify a rate of change. ATTACK:Ø selects a very slow increase rate, and ATTACK:65535 selects a very fast increase rate. (Actually, 1 is very slow. Ø is stopped, or no change.) A blank song created with the NEW command contains some envelope settings which are useful for testing songs. Usually you enter the basic notes of a song, play around with the tempo (playback speed) if necessary using SPEED commands and/or different QUARTER settings, and once you're satisfied with the tempo you go on to the envelope settings. This is because the SPEED command doesn't change any of the envelope settings. If you perfected your envelope settings and then used a SPEED command, the envelopes would no longer be perfect. This is needlessly complex to correct, so it is best to get the tempo going right before starting in on envelopes.

To change the initial envelope settings, just position the cursor at the appropriate setting and type in a new value. For example, if you wish to have a slower attack rate, you might position the cursor at the ATTACK 8192 and type ATTACK:78ØØ. Few songs use the same envelopes on all parts or even the same envelope throughout any particular part. At any point in a part, you can just "insert" new envelope parameters. During playback, the most recent setting (for each part) is used for envelope production. Since there are notes (and rests) between one envelope specification and another, the playback routines will not "see" the later specifications in the part until the note before them is finished. When they finish a note, they look at the next thing in the part. If it's not a note or a rest, they make whatever change is requested (a new attack value, for example) and then continue with the next thing in the part (until a note or rest is finally found).

GAP is not mentioned in the INTRODUCTION program. Further, the INTRODUCTION program claims that the sustain stage of the envelope lasts "as long as desired". Usually, on a synthesizer or a piano, the sustain stage ends (and the release stage begins) whenever the key being pressed is released (hence the word "release", obviously). There aren't any keys to release in the music data. So, the GAP function is used. It is used to specify how long before the next note begins the release stage should begin. For example, using QUARTER:24Ø settings, a whole note (96Ø time periods) played with a GAP setting of 24Ø would have three quarter notes (96Ø-24Ø, or 72Ø time periods) worth of attack, decay, and sustain; then one quarter note (24Ø time periods) worth of release. A rest automatically starts the release stage if it wasn't already. Notes shorter than the GAP setting have no release stage unless followed by a rest. GAP:65535 is used when no automatic release stage is desired.

Now is the time for all good men to experiment with envelope settings. Don't come back to this manual without experimenting for at least 7 million time periods.

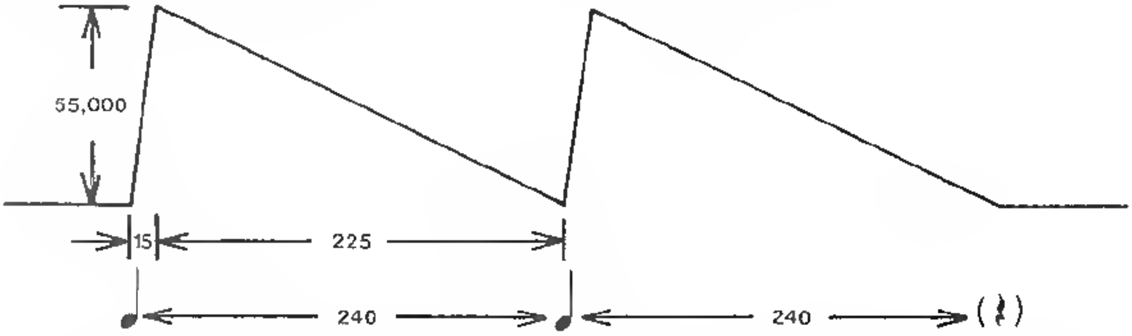
You are now ready for the serious explanation of envelope production. Although theories change from time to time, today's leading scientists in enveology agree on the "wandering loudness" explanation. This one seems to fit the reality of the synthesizer most closely. The two main ingredients of this are "current loudness" and "desired loudness". The current loudness refers to a number which ranges from Ø to 65535. This number divided by 256 is the actual volume setting on the synthesizer at the moment. The desired loudness is also a number from Ø to 65535. The current loudness is "attracted" to the desired loudness, so it attempts to get closer and closer to it. Once each time period, the current loudness can increase by an amount less than or equal to the attack setting, or it can decrease by an amount less than or equal to the "current decay" setting. (Not to be confused with the "decay setting".) In this fashion, it will arrive at the desired loudness as quickly as the attack/current decay settings permit. Once the current loudness collides with the desired loudness, the desired loudness spontaneously changes to a new value, called the "current sustain level" (not to be confused with the "sustain setting"). Probability states that the new desired loudness may be different than the current loudness (although the current loudness is equal to the old desired loudness), so the current loudness must again seek the desired loudness. This astounding natural process continues at all times during playback. The current loudness cannot be affected directly, so it must be "guided" by selecting appropriate parameter settings.

Notetrinos generated using a high-power paramatron at the University of Northern South Dakota (just across the border from Hoople) have revealed the following characteristics of these settings. (What?) When a new note begins, the most recent decay setting is written into the "current decay" rate, the most recent volume setting is written into the "desired loudness", and the most recent sustain setting is written into the "current sustain". This causes the attack and decay stages of the envelope to occur, since the current loudness (and thus the synthesizer volume) will raise (at the attack rate) to the selected volume level, at which time the sustain level becomes the new desired loudness, causing the current loudness to drop to the sustain level (at the decay rate). Once the sustain level is reached, the desired loudness stays constant (since it is equal to the current sustain setting which would normally become the new desired loudness) and thus the sustain stage of the envelope occurs until something changes.

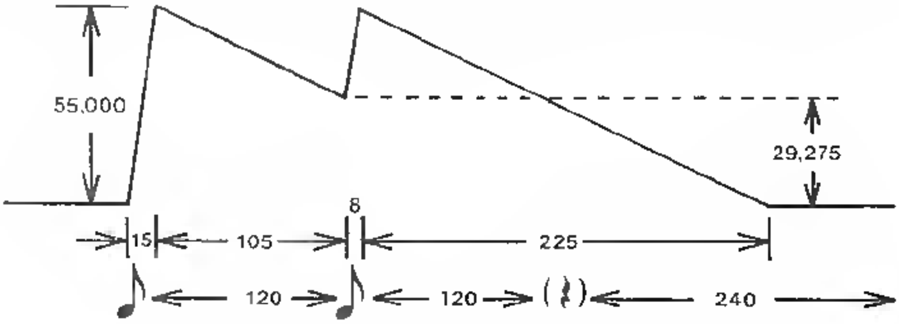
Something changes when either (a) the time remaining for the current note equals the most recent GAP setting, (b) a rest is encountered, or (c) a new note is encountered. Case (c) has already been discussed (above). In either case (a) or (b), the release stage must begin. This is done by writing the most recent release setting into the "current decay" and a zero into the "desired loudness" and "current sustain". The current loudness (and, again, thus the actual synthesizer volume) then naturally drops to zero at the selected release rate.

This simple process generates a variety of complex envelopes, for single notes or for several. Be ye not confused: each note does not necessarily have an "attack" and "decay" stage (and so forth). In fact, if the current loudness is greater than the latest volume level when a new note begins (for example, the volume setting was just lowered drastically before this note, and the previous note had been at a very high volume with too slow a decay/release rate to drop very far), the note would begin with a "decay" stage, since the current loudness would have to go down to intercept the desired loudness (which would be the new volume level). Thus, the envelope parameters are not limited to a single note. In general, however, one will arrange the parameters so the envelope will be limited to a single note.

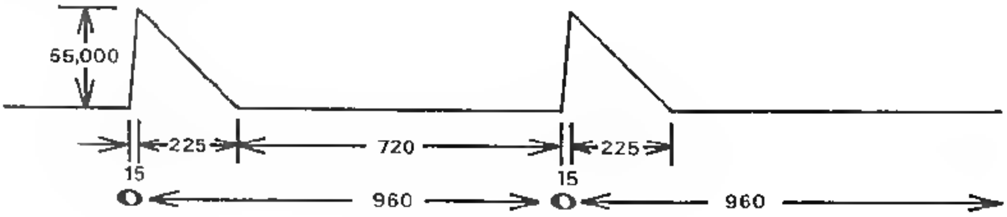
Some examples are in order. Let's say we want a simple AD (attack-decay, or "ping") envelope with a volume level of 55ØØØ. Further, let's say it is a quarter note with standard QUARTER settings (24Ø time periods) and we want the first 16th of the note to be the attack stage, and the remaining 15/16ths to be a full decay. The attack rate will have to be designed to take the current loudness from Ø to 55ØØØ in 24Ø/16 time periods. 55ØØØ/(24Ø/16) is 3666.67 so we want an attack setting of 3667. The decay rate will have to take the current loudness from this peak of 55ØØØ back down to Ø in 24Ø*15/16 time periods. 55ØØØ/(24Ø*15/16) is 244.44 so we want a decay setting of 245. The loudness contour will appear thus:



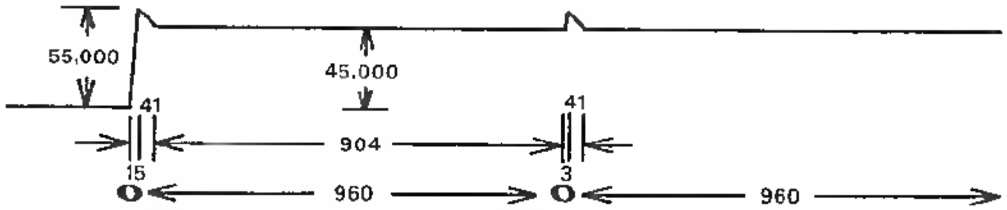
The GAP setting must be 65535 to avoid a release stage. Now, what if we played an eighth note with this setting? The loudness contour would appear thus:



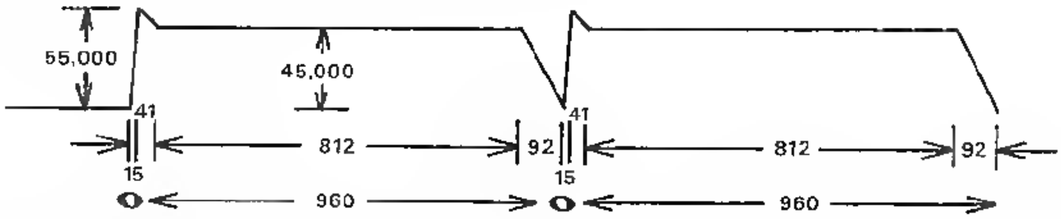
If an eighth note is followed by a rest, the release stage will begin. Therefore the release setting should be set to the same as the decay setting, unless you want something different to happen on notes followed by rests. What if we played a whole note? Behold:



This assumes the sustain level was set to Ø. What if it were 45ØØØ?:



This is almost an ADSR (attack-decay-sustain-release) envelope. All we need is release. Let's say we want it to take half as long to release as the quarter note example took to decay. That means we'll need a release rate which is twice as fast, or 2*245 which is RELEASE:49Ø. Now, it will take 45ØØØ/49Ø time periods for the current loudness to drop from 45ØØØ (the sustain level) to Ø, so we need a GAP setting of 45ØØØ/49Ø (which is 92) or greater if we want the release to go clear down to zero. That looks like this:

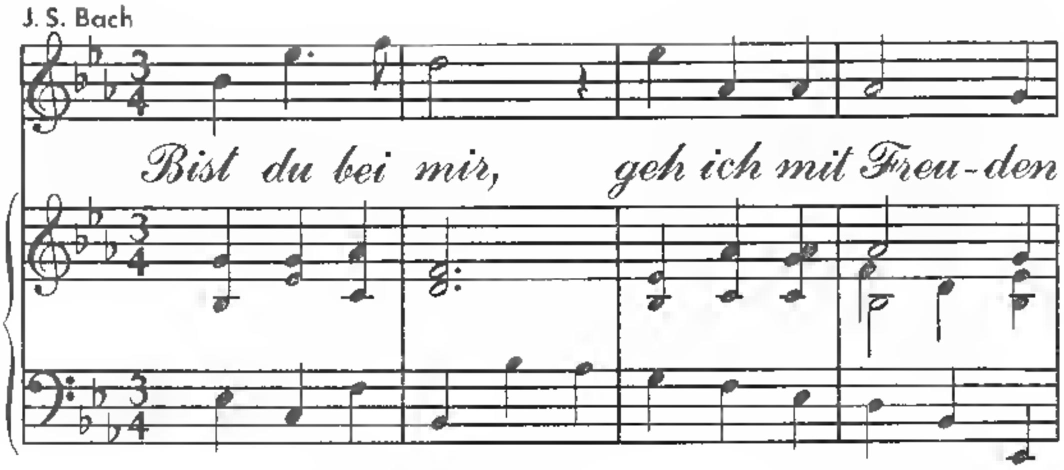


The sustain level need not be less than the volume level. For example, with a sustain level equal to the volume level, you get an attack-sustain-release envelope (organ like, using fast attack and release rates).

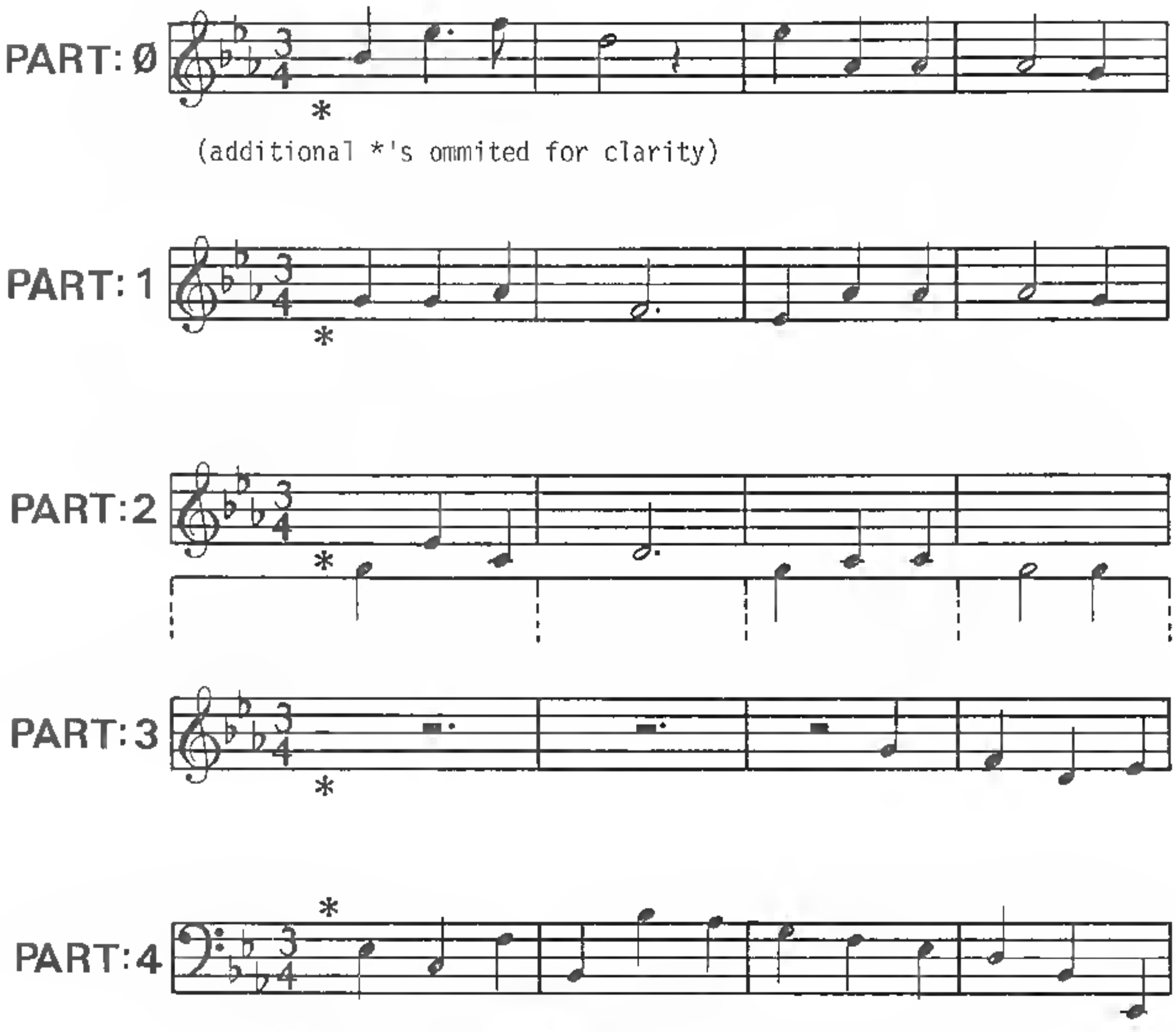
Experiment more with the settings. Draw graphs like the ones above if they help you. Look at other people's envelope settings if you run out of ideas. Here's a real tip: program what would normally be a whole part into a subroutine instead. Then you can call it from two parts, and use different envelope settings on each part (don't put envelope settings in the subroutine!). This will let you make more complex sounds, especially using different transpose settings or by putting a short rest before the CALL in one of the parts to delay it slightly (for an "echo" effect) or both.

## RECOMMENDED READING

For those of you who are unfamiliar with standard sheet music notation, or for those who encounter some particularly obscure notation, there is an excellent book which you can order from any bookstore. Just ask your local store to order "Music Notation, A Manual of Modern Practice" by Gardner Read, Taplinger Publishing Co. ISBN Ø-8ØØ8-5453-5. In the unlikely event that you have no local bookstores, you can order it from ALF (part number 11-2-1).



BECOMES:



(additional *'s ommited for clarity)

SAMPLE SONG BREAKDOWN

## SUMMARY OF COMMANDS

ENTRY has four types of commands. They are:

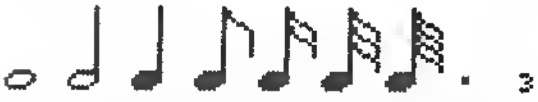
1. Commands which are done immediately and have no effect on the song data.
2. Commands which are done immediately and have an effect on the song data.
3. Commands which are stored in the song data and do not affect playback directly.
4. Commands which are stored in the song data and do affect playback directly.

All commands, except those entered using the paddles, are typed in using the Apple keyboard in the following fashion. Each command has a "keyword", for example NEW or VOLUME. Some commands have one or more parameters, in which case the keyword is followed by a colon (:) and the parameter, for example VOLUME:55ØØØ. Thus, a command is always entered by typing the keyword and pressing return; or by typing the keyword, a colon, one or more parameters, and pressing return. (Do not type any spaces.) Since the keyword is always followed by a return or a colon, ENTRY has been written to allow abbreviation of the keyword. You can shorten any keyword as much as you like, as long as there are still enough letters to tell it apart from any other keyword. For example, INTEGER can be shortened to just I since no other keyword starts with I. SUBROUTINE can be shortened to SUB, but not to SU since it could then be either SUBROUTINE or SUSTAIN. An example of a complete abbreviated command is SUB:Ø instead of SUBROUTINE:Ø. The right and left arrows on the Apple keyboard can be used to backspace and to forward space for error correction. When return is pressed, only letters to the left of the flashing cursor are considered part of the command, other letters are ignored. Control X can be used to clear the line and start over.

In the bold type for each command, anything inside <broken brackets> is an explanation rather than something to be typed literally. Anything inside [brackets] is optional.

## TYPE 1 COMMANDS

These commands are done immediately. The song data is not changed at all.



The seven note duration symbols, plus "." and "3", are used to select a new note entry duration. (See REST and PADDLE 1 under Type 4 Commands.) They are requested by pressing Paddle Ø's button while the upward-pointing arrow is aiming at the desired symbol. When one of the seven note duration symbols is requested, a block is lit under it. All other blocks under note duration symbols (including "." and "3") are turned off. When "." is requested, the block under it changes (becomes lit if it wasn't, or is cleared if it was lit). When "3" is requested, the block under it changes.

**♯ ♭ ♮**

The three accidental control symbols are used to select accidental control for future note entry (see PADDLE 1 under Type 4 Commands). They are requested by pressing Paddle Ø's button while the upward-pointing arrow is aimed at the desired symbol. When one of the accidental control symbols is requested, the block under it is changed (becomes lit if it wasn't, or is cleared if it was lit) and the blocks under the other two accidental control symbols are cleared.

**→ ←**

The left and right movement controls are used to move the cursor left or right. They are requested by pressing Paddle Ø's button while the upward-pointing arrow is aimed at the desired symbol. When one of the movement control symbols is requested, the cursor will move one item in the indicated direction. Movement to the left of the first item in a subroutine or part is not allowed. Movement to the right of the end marker in a subroutine or part is not allowed. When a movement is requested which is not allowed, the request is ignored and the Apple speaker will beep.

**INS**

The insert symbol is used to turn insert mode on or off. It is requested by pressing Paddle Ø's button while the upward-pointing arrow is aimed at INS. When requested, the block under INS is changed (becomes lit if it wasn't, or is cleared if it was lit). "Insert mode" is on when the block under INS is lit, or when the cursor is at the end marker of a part or subroutine. All Type 3 and Type 4 Commands are affected by insert mode.

**▷**
**→**

The speaker/arrow symbol is used to select playback during forward (right) movement. It is requested by pressing Paddle Ø's button while the upward-pointing arrow is aimed at the speaker/arrow symbol. When requested, the block under the symbol is changed (becomes lit if it wasn't, or is cleared if it was lit). When lit, notes moved past with the right movement symbol, and notes deleted with the DEL symbol, are sounded through the synthesizer.

**GOTO:<Ø-8>**

The GOTO command is equivalent to the PART command (a Type 1 Command) except that a MEASURE command (a Type 1 Command) is automatically performed after the indicated part has been selected. The measure number used for the MEASURE command is whatever measure number was displayed on the screen at the time the GOTO command was entered. Sample command: GOTO:1 (return).

**INTEGER**
The INTEGER command is used to exit ENTRY and return to BASIC. The current song data is lost. ENTRY cannot be run again without first being reloaded. Note that when using the APPLESOFT version, the INTEGER command is used to return to BASIC, but APPLESOFT BASIC will be returned to rather than Integer BASIC.

**LENGTH:<Ø-65535>**
The LENGTH command is used to select a non-standard note duration. (See PADDLE Ø and PADDLE 1 under Type 4 Commands.) When entered, all blocks under the seven note duration symbols and under "." and "3" are cleared. The indicated duration is saved for future note and rest entry use. Sample command: LENGTH:48 (return).

**MEASURE:<Ø-65535>**
The MEASURE command is used to view a particular measure within a part or subroutine. The cursor moves to the first item within the specified measure number. MEASURE:Ø is equivalent to MEASURE:1. If no such measure exists, the cursor is moved to the end marker of the part or subroutine. Sample command: MEASURE:249 (return).

**PART:<Ø-8>**
The PART command is used to view a particular part (and thus select that part for possible editing). The cursor moves to the first item in the selected part, or to the end marker for that part if there are no items in the part. Sample command: PART:1 (return).

**PLAY[:F]**
The PLAY command is used to perform the current song (using a modified version of the PERFORM program). A simple low-res color display is shown during playback. In this display, each part has a blue horizontal line. In this line is a yellow dot which marks the position of middle C for that part (this dot will not be present when playing very high pitched notes). This middle C marker slides left and right one or more octaves if necessary to show whatever pitch range is currently being used. Above the horizontal line, a block is shown which indicates the pitch being produced. Higher pitches are to the right of the display. The color of this block indicates the approximate "current loudness" of the pitch as follows: Ø-4Ø95 black, 4Ø96-8191 magenta, 8192-12287 dark blue, 12288-16383 purple, 16384-2Ø479 dark green, 2Ø48Ø-24575 grey, 24576-28671 medium

blue, 28672-32767 light blue, 32768-36863 brown, 36864-40959 orange, 40960-45055 grey, 45056-49151 pink, 49152-53247 green, 53248-57343 yellow, 57344-61439 aqua, 61440-65535 white (loudest). (Based on Apple's suggested color names; actual colors may vary.) Ignoring the fact that there are two colors named grey, each color represents any of 16 different actual volume settings on the synthesizer, since there are only 16 colors for 256 settings. PLAY:F performs the current song using the PERFORM program (that is, with no display). NOTE: both PLAY commands change (a) the CHANNEL function settings and (b) the subroutine FE bytes. These changes will not be apparent to the ENTRY user, but could affect PERFORM users. See the PERFORM section for additional information. Sample command: PLAY (return).

**SAVE[:<song name>[<disk specifications>]]**
The SAVE command is used to write the current song data on cassette tape (or whatever might be connected to the Apple's cassette output jack) or on disk. SAVE saves the song to cassette tape. SAVE:<song name>[<disk specifications>] saves the song to disk. Both commands are used in the same fashion as the SAVE commands in BASIC. One exception: song names may contain Ø to 28 characters, including any character except comma (for any character, including the first); control characters and trailing spaces are ignored, but leading spaces are not. Sample command: SAVE:GALACTIC TRIUMPH,D2 (return).

***DISK[:<comment>]
The ***DISK command increases the karma of the user when using DOS 3.1. This command has no effect when using DOS 3.2 or a cassette based system. Sample command: ***DISK: FILE NOT FOUND ERROR (return).

## TYPE 2 COMMANDS

These commands are done immediately. They do not cause an item to be written at the current cursor location, as Type 3 and Type 4 Commands do, but they do affect the current song data.

**DEL**
The DEL symbol is used to delete the item the cursor is currently at. It is requested by pressing Paddle Ø's button while the upward-pointing arrow is aimed at DEL. When requested, the item the cursor is at is deleted from the song data. If it is a note, it is sounded through the synthesizer if the speaker/arrow block is lit (see the speaker/arrow Type 1 Command). The end marker of a part or subroutine cannot be deleted. If this is attempted, the Apple speaker beeps.

**DELETE:<1-255>**
The DELETE command is used to remove one or more items from the current part

or subroutine. It is the same as one or more DEL symbol requests (above) except the notes are never sounded and there is no "beep" when an attempt is made to delete the end marker. The number of DEL's is selected by the <1-255> parameter. More than 255 items can be deleted only using more than one DELETE command. Sample command: DELETE:73 (return).

**EDIT**

The EDIT command is used to increase the number of parts, change the suggested speed, and/or change any or all of the 4 title lines. Once entered, the command proceeds to ask for the new NUMBER OF PARTS?, SUGGESTED SPEED?, and TITLE LINE 1 through TITLE LINE 4. If there is no change desired on any item, just press return. Otherwise, enter the new value and press return. For each TITLE LINE, the current line is displayed and can then be edited using the left and right arrow keys on the Apple keyboard. Note that when return is pressed for a title line, all characters to the right of the flashing cursor, and the character under the flashing cursor unless it is the 4Øth character, are set to space. The SUGGESTED SPEED must be from Ø to 255. (I through 255 select paddle speeds, and Ø activates Timing Mode.) The NUMBER OF PARTS? must be greater than or equal to the current number of parts, but less than 1Ø. (Remember you can only play 3 parts per synthesizer, and 1 less part when using Timing Mode.) If the number of parts is increased, the stereo settings are set to standard settings (see NEW, a Type 2 Command; and STEREO, a Type 2 Command). See SUBROUTINE (a Type 2 Command) for details on reduction of "notes free" when increasing the number of parts. The cursor is set to the first item in Part Ø. Sample command: EDIT (return).

**LOAD[:<song name>[<disk specifications]]**

The LOAD command is used to load a song from cassette tape (or whatever is connected to the Apple's cassette in jack) or disk. The song currently in memory is lost. These commands are used the same as the LOAD commands in BASIC. See SAVE (a Type 1 Command) for additional comments. The cursor is set to the first item in Part Ø. Sample command: LOAD:GALACTIC TRIUMPH (return).

**NEW**

The NEW command is used to start fresh. Once entered, the NEW command asks for the NUMBER OF PARTS? which should generally be entered as I. If return is pressed, 1 is assumed. The number of parts cannot exceed 9. Remember that parts created cannot be destroyed and that song playback ends when the end of the highest numbered part is reached. New parts (created either with NEW or with EDIT, a Type 2 Command) contain KEY:C, TIME:4/4, QUARTER:24Ø, GAP:65535, TRANSPOSE:Ø, ATTACK:8192, DECAY:5Ø, VOLUME:55ØØØ, SUSTAIN:Ø, and RELEASE:5Ø. (All subroutines and parts always end with an end marker.) Stereo is set to the standard values: STEREO:2,LRLRLR and STEREO:3,MLRMLRMLR. The NEW command then asks for the SUGGESTED SPEED? which can be given as any integer from Ø to 255, or just press return for 255. Finally, the NEW command asks for the 4 TITLE LINEs. These are initially set to all spaces. The cursor is set to the first item in Part Ø. Sample command: NEW (return).

**SPEED:<1-65535>[/<1-65535>]**

The SPEED command is used to change the duration of all notes, rests, and QUARTER functions in all parts and subroutines. The colon after SPEED is followed by an integer from 1 to 65535 to multiply all time durations by. This is optionally followed by a slash (/) and another integer from 1 to 65535 indicating a number to divide by. (If not specified, this is assumed to be 1.) All time durations are multiplied by the first integer, then divided by the second integer. Any "remainder" (or non-integral portion) is ignored, and the result MOD 65536 is used. For example, a note length of 24Ø divided by 5Ø (using SPEED:1/5Ø) would become 4 since 24Ø/5Ø equals 4.8. The .8 time periods dropped will eventually accumulate (differently in different parts) and create unusual timing. Therefore, such non-integral results should usually be avoided. Any Ø results are changed to 1. **CAUTION**: extreme care must be taken to avoid destruction of the song! Saving the song prior to attempting a SPEED command is strongly recommended. Sample command: SAVE:GALACTIC TRIUMPH (return) SPEED:1/2 (return).

**STEREO:<2-3>,<string>**

The STEREO command is used to change the stereo selection programmed in the song. Although stereo outputs are available only when using two or three synthesizers, you may wish to set the stereo selection even when using only one unit if the song may be played by others having more units. STEREO:2,<string> sets the stereo which will be used when the song is played back on a system with 2 synthesizers. It applies only to songs having 6 or fewer parts. The <string> must consist of L's (for Left) and R's (for Right). There should be one letter for each part. The first letter specifies the position for Part Ø, the second for Part 1, etc. There cannot be more than 3 L's or more than 3 R's. Note that songs should usually not have more than 2 R's. If a song has 3 R's, it cannot be played on a system with Timing Mode unless 3 synthesizers are used. STEREO:3,<string> sets the stereo which will be used when the song is played back on a system with 3 synthesizers. It is used the same as the STEREO:2,<string> command except that in addition to L's and R's, M's can be used (for Middle). There cannot be more than 3 M's, and no more than 2 M's can be used if Timing Mode is to be used during playback. When creating a song for general use, STEREO:3,<string> should always be specified. STEREO:2,<string> should also be specified on all songs having 6 or fewer parts. **NOTE**: the EDIT command changes both the STEREO:2,<string> and STEREO:3,<string> settings if the number of parts is increased. The stereo settings selected are programmed into the CHANNEL function (see the PERFORM section) and thus will be saved with the song. Sample command: STEREO:2,LLR (return).

**SUBROUTINE:<Ø-99>**

The SUBROUTINE command is used to create a subroutine, or to view (and thus ready for editing) an existing subroutine. (Note: this command may be considered a Type 1 Command if used to access an existing subroutine rather than create a new one.) The creation of a new subroutine will reduce the number of free notes by the following amounts depending on the number of parts: 2 for 1 part, 3 for 2, 4 for 3 or 4, 5 for 5, 6 for 6 or 7, 7 for 8, and 8 notes for 9 parts. (NOTE: increasing the number of parts with EDIT, a Type 2 Command, reduces the number of free notes by enough to account for the difference in storage requirements for each subroutine (since more "notes" of storage are required per subroutine when more parts are present, as shown above), plus 12 and 2/3rds notes per new part.) The cursor is positioned to the first item in the selected subroutine, or the end marker in that subroutine if there are no items. **CAUTION:** subroutines are assigned numbers from Ø up (by ones) when a song is loaded and when RESET is pressed (CØØG must be typed on systems without an Auto-Start ROM). The numerical order of the subroutines does not change. Sample command: SUBROUTINE:83 (return).

## TYPE 3 COMMANDS

These commands are not done immediately, but rather are stored in the song data at the current cursor position. The item currently at the cursor position is erased unless insert mode is on. These commands do not affect playback. They affect only newly entered notes and rests, or the screen display. Commands of this type included within a subroutine affect only the display and entry of notes within the subroutine itself, and not within any part (or other subroutine) calling the subroutine. The number of notes free goes down by 1 for each inserted command, but stays the same for replaced commands.

**KEY:<1-6><S-F> or KEY:C**

The KEY command is used to change the key signature. (If no KEY command has occured in the part or subroutine so far, the key is assumed to be KEY:C.) KEY:C specifies no sharps or flats, and an integer from 1 to 6 followed by an S or an F specifies the indicated number of sharps (S) or flats (F). All notes entered so as to appear in the song data after this KEY command (but before the next KEY command) will be affected by this KEY command. Any note not entered as "sharp", "flat", or "natural" will be changed to sharp if it is one of the notes indicated as sharp in the key signature, or changed to flat if it is one of the notes indicated as flat in the key signature. Notes not indicated as either sharp or flat by the key signature are left as is. Sample command: KEY:3S (return).

**QUARTER:<1-65535>**
The QUARTER command is used to change the duration of notes entered except when using non-standard durations with LENGTH (a Type 1 Command). All notes entered so as to appear in the song data after this QUARTER command but before the next QUARTER command will be affected. (If no QUARTER command has occured in the part or subroutine so far, it is assumed to be QUARTER:24Ø.) See the PADDLE Ø and PADDLE 1 Type 4 Commands for additional details. Sample command: QUARTER:48Ø (return).

**TIME:<1-19>/<note>**
The TIME command is used to change the time signature. (If no TIME command has occured in the part or subroutine so far, the meter is assumed to be 4/4.) The colon after TIME is followed by the number of notes (of a certain duration) to occur per measure. This is followed by a slash (/) which does not mean division (this is a special case). The slash is followed by an integer which specifies the note duration referenced by the other integer. It must be 1 for a whole note, 2 for a half, 4 for a quarter, 8 for an eighth, or 16 for a sixteenth note. The number of time periods allowed per measure will be the current QUARTER setting times 4 times the number before the slash, all divided by the number after the slash. This command determines the positioning of measure bars, which in turn affects whether a note is sharp (or flat) or not (see the PADDLE 1 Type 4 Command). It affects all notes entered so as to appear in the song data after this TIME command but before the next TIME command. Sample command: TIME:2/2 (return).

## TYPE 4 COMMANDS

These commands are not done immediately, but rather are stored in the song data at the current cursor position. The item currently at the cursor position is erased unless insert mode is on. These commands are executed during playback. They are executed during a subroutine call and thus may effect notes entered in a given part (or subroutine) after a call to the subroutine containing these commands. The number of notes remaining goes down by 1 for each inserted command, and stays the same for replaced commands, except as noted for TIE. **<value>** always refers to an integer from Ø to 65535, optionally followed by a slash (/) and another integer from Ø to 65535. When the slash is specified, the indicated division is done and the resultant value (ignoring any remainder or non-integral portion) is used as the parameter.

**REST**
The REST symbol is requested by pressing Paddle Ø's button while the upward-pointing arrow is pointing at REST. When requested, a rest is written in the

song data. The duration of the rest is determined in the same fashion as the PADDLE 1 Type 4 Command (below).

**PADDLE 1**

Note entry is accomplished by pressing Paddle 1's button. The vertical position of the note cursor (controlled by Paddle 1's knob) determines the pitch of the note, subject to various sharps and flats, and (during playback only) the current TRANSPOSE (Type 4 Command) setting. Notes will be natural, sharp, or flat; as indicated by a block under one of these in the menu, and the blocks cleared, if one of these blocks is lit. Otherwise, notes are entered as natural unless they must be sharp or flat due to the current key signature or due to a prior note in the measure of the same pitch being sharp or flat. (Note: all octaves are affected by the key signature, but not by prior sharp or flat notes in the measure.) Natural, sharp, or flat signs are displayed on the screen only when necessary. Duration is as specified by LENGTH (a Type 1 Command) unless one or more blocks are lit under the seven notes in the menu. (Note: "." and "3" do affect LENGTH settings.) If a block is lit, the length will be assumed to be as specified by the most recent QUARTER command for quarter notes, and proportional values for all other notes. A block under "." multiplies the length by 3/2, and a block under "3" multiplies the length by 2/3. (A block under both multiplies the length by 2/3 and then by 3/2.) Entry of a sixty-fourth note (selected by a block under the sixty-fourth note) is not allowed if the "." block is lit. (Dotted sixty-fourth notes are never displayed.)

**TIE**

The TIE symbol is requested by pressing Paddle Ø's button while the upward-pointing arrow is pointing at TIE. When requested, the duration which would be used if a note were entered (see the PADDLE 1 Type 4 Command) is added to the duration of the note or rest the cursor is currently at. (If the cursor is not at a note or rest, the Apple speaker beeps and the cursor moves left one item.) This command is unaffected by insert mode, and it never changes the number of notes free.

**ATTACK:<value>**

The ATTACK command changes the current attack setting. The value specified is the maximum amount the "current loudness" can increase in any given "time period". Sample command: ATTACK:55ØØØ/3Ø (return).

**CALL:<Ø-99>**

The CALL command is used to have the Type 4 Commands in the specified subroutine be executed during playback. The integer (from Ø to 99) specifies which subroutine should be done. More than one part may call the same subroutine (or different subroutines) at the same time. A subroutine may call itself provided at least one time period of duration occurs within the subroutine prior to the call to itself. A CALL cannot be entered until after its subroutine has been created. See SUBROUTINE (a Type 2 Command) for additional information. Sample command: CALL:83 (return).

**DECAY:<value>**
The DECAY command changes the current "decay setting". The value specified is the maximum amount the "current loudness" can decrease in any given "time period" unless the RELEASE rate is currently being used. Sample command: DECAY:1ØØ (return).

**GAP:<value>**
The GAP command changes the current gap setting. When the time remaining for any note equals the current gap setting, the release stage of the envelope begins. Sample command: GAP:6Ø (return).

**POKE:<Ø-255>,<Ø-255>,<Ø-255>**
The POKE command is used to enter non-standard commands. **CAUTION:** use of this command renders this documentation meaningless and may well scramble memory during playback. Integers from Ø to 191 followed by Ø and Ø (for example, POKE:78,Ø,Ø) enter notes of zero duration, the correct duration can be TIEd in. For information on other values, see the PERFORM section, and the SONG DATA FORMAT heading in this section. Sample command: POKE:Ø,24Ø,Ø (return).

**RELEASE:<value>**
The RELEASE command changes the current release setting. The value specified is the maximum amount the "current loudness" can decrease in any given "time period" unless the DECAY rate is currently being used. Sample command: RELEASE:1ØØ (return).

**SUSTAIN:<value>**
The SUSTAIN command changes the current "sustain setting". The value specified is the "desired loudness" which the "current loudness" follows, unless the desired loudness is currently Ø for a release stage or the current volume setting for an attack stage. Sample command: SUSTAIN:45ØØØ (return).

**TEMPO:<value>**
The TEMPO command is used to change the playback tempo. It need appear in only one part since it affects the playback speed (tempo) of all parts. Although it should be included in any song for general use, it is active only when using Timing Mode (see the TIMING MODE section). The TEMPO setting should be about 19.25*(<paddle setting>+1). There will be 1782ØØØ/TEMPO time periods per second, unless the selected time period is too short for all necessary computations to occur. Sample command: TEMPO:4735 (return).

**TRANSPOSE:<Ø-255>**

The TRANSPOSE command is used to change the current transpose setting. Values from Ø to 127 raise all following pitches (until the next TRANSPOSE command) by Ø to 127 quarter steps; values form 255 to 128 lower all following pitches by 1 to 128 quarter steps. 24 quarter steps equals 1 octave. Sample command: TRANSPOSE:232 (return).

**VOLUME:<value>**

The VOLUME command changes the current volume setting. The value specified is the "desired loudness" which the "current loudness" follows unless the envelope is not currently in an attack stage. Sample command: VOLUME:5ØØØØ (return).

# TIPS

## PARTIAL STARTING MEASURE

Often songs begin with a measure which is short, perhaps containing only a single note. If such a song were entered in the normal fashion, the measure bars would not appear at the correct places. There are many ways of solving this problem. The simplest and perhaps best way is to start by entering a rest which is long enough to fill one measure when the partial (starting) measure is entered after the rest. Not only does this put the measure bars in the right places, it also causes a brief delay before song playback begins during a PLAY command, which may be considered desirable. Another method is to put the partial measure in a subroutine, and call it. (The duration of notes within a subroutine is not added to a part which contains a CALL to that subroutine.) Yet another method is to enter the partial measure, and then enter a TIME or a QUARTER command to start the measure over.

## RESTS AT THE END OF PARTS

Each part should end with a rest. It can be as short as you like, and it serves to begin the release stage of the envelope. Otherwise a release stage may begin unexpectedly (when the constantly cycling time remaining equals the current GAP size). Additionally, the highest numbered part should end with a rest long enough to let all parts decay (or release, actually) down to zero volume, and perhaps even show a "blank" screen for a second. PERFORM users may find this particularly necessary, lest the parts continue playing after PERFORM returns to the calling program.

## PADDLE SETTINGS

Paddle settings which are too small will create "time periods" which are not long enough for all necessary calculations. When this happens, the "time period" is lengthened so that all calculations are completed. Since the calculation time required varies, the song playback speed will vary too. There is no time period variation when the paddle setting is high enough. Generally, paddle settings lower than 15Ø are never used. Songs having many parts active and using several levels of subroutines may require even higher settings. The number of time periods in one second is approximately 93ØØØ/(<paddle setting>+1).

## "BACK-UP"

While entering particularly long songs, it is a good idea to save the song periodically in case the power fails, ENTRY hits an undiscovered bug, or you accidently delete half the melody.

### TRANSPOSE

Each part must contain a TRANPOSE before the first note, even if it is a TRANSPOSE:Ø.

### COPYING SONGS WITHOUT ENTRY

Systems equipped with Integer BASIC can copy songs from one tape or disk to another without running ENTRY. Just load the song as if it really were an Integer BASIC program, and save it. Since it isn't a BASIC program, attempting to change or delete a line, or attempting to RUN it, would probably scramble the song data; however, a load followed immediately by a save will work properly.

### RESET

On systems without an Auto-Start ROM, CØØG (return) must be typed if RESET is pressed. That's C zero zero G, not COOG. RESET can safely be used during a PLAY command. RESET must not be used during the execution of any other command, or the song data may be destroyed.

### INTEGER/APPLESOFT SWITCH

On systems with a ROM card (for Applesoft or Integer BASIC), the switch must be set for a start-up language which matches the version of ENTRY being used.

## SONG DATA FORMAT

Song data is stored as described in the PERFORM section with the following changes:

1. Song data always begins in memory at 5ØØØ hex.
2. The END command (FF ØØ ØØ) is followed by a byte giving the suggested speed, then 16Ø bytes which form the four title lines.
3. The QUARTER command is stored with command type FB hex.
4. The KEY command is stored with command type FC hex. A parameter of zero indicates C. Otherwise, the number of sharps/flats is stored with the most significant bit being Ø for flat or 1 for sharp. The third byte is not used.
5. The TIME command is stored with command type FD hex. The second byte indicates the number of notes per measure, and the third byte the type of note.
6. All TRANSPOSE commands have a third byte of FE. This allows the least significant bit of each note to indicate sharp or flat.
7. When loaded using Integer BASIC, locations CA and CB hex ("PP") indicate the starting address of the data. Locations 4C and 4D hex ("HIMEM") indicate the address past the last byte of data.

## SELECTED HEX ADDRESSES

4C & 4D: defines the address of the first byte of unavailable memory
72: defines the lowest slot number times 16
87: defines the number of synthesizer units
5E & 5F: defines the address of the first byte following the song's title lines (end of song pointer)
5ØØØ: start of song data
A76: start of pitch divisor table
4F38: start of Entry-generated subroutine address table
4D52-4D75: part initialization data
4ECD-4F36: command table expansion area
4DAA-4DB2: standard stereo positions
Base page usage: (see also PERFORM base page usage)

```
    Ø-19   26-27   36-39   3C-3F   4A-4D   5Ø-55   58-8F   CA-CD
```

### TRANSPOSE

Each part must contain a TRANPOSE before the first note, even if it is a TRANSPOSE:Ø.

### COPYING SONGS WITHOUT ENTRY

Systems equipped with Integer BASIC can copy songs from one tape or disk to another without running ENTRY. Just load the song as if it really were an Integer BASIC program, and save it. Since it isn't a BASIC program, attempting to change or delete a line, or attempting to RUN it, would probably scramble the song data; however, a load followed immediately by a save will work properly.

### RESET

On systems without an Auto-Start ROM, CØØG (return) must be typed if RESET is pressed. That's C zero zero G, not COOG. RESET can safely be used during a PLAY command. RESET must not be used during the execution of any other command, or the song data may be destroyed.

### INTEGER/APPLESOFT SWITCH

On systems with a ROM card (for Applesoft or Integer BASIC), the switch must be set for a start-up language which matches the version of ENTRY being used.

## SONG DATA FORMAT

Song data is stored as described in the PERFORM section with the following changes:

1. Song data always begins in memory at 5ØØØ hex.
2. The END command (FF ØØ ØØ) is followed by a byte giving the suggested speed, then 16Ø bytes which form the four title lines.
3. The QUARTER command is stored with command type FB hex.
4. The KEY command is stored with command type FC hex. A parameter of zero indicates C. Otherwise, the number of sharps/flats is stored with the most significant bit being Ø for flat or 1 for sharp. The third byte is not used.
5. The TIME command is stored with command type FD hex. The second byte indicates the number of notes per measure, and the third byte the type of note.
6. All TRANSPOSE commands have a third byte of FE. This allows the least significant bit of each note to indicate sharp or flat.
7. When loaded using Integer BASIC, locations CA and CB hex ("PP") indicate the starting address of the data. Locations 4C and 4D hex ("HIMEM") indicate the address past the last byte of data.

## SELECTED HEX ADDRESSES

4C & 4D: defines the address of the first byte of unavailable memory

72: defines the lowest slot number times 16

87: defines the number of synthesizer units

5E & 5F: defines the address of the first byte following the song's title lines (end of song pointer)

5ØØØ: start of song data

A76: start of pitch divisor table

4F38: start of Entry-generated subroutine address table

4D52-4D75: part initialization data

4ECD-4F36: command table expansion area

4DAA-4DB2: standard stereo positions

Base page usage: (see also PERFORM base page usage)

```
    Ø-19   26-27   36-39   3C-3F   4A-4D   5Ø-55   58-8F   CA-CD
```

# 4

# PLAY

The PLAY program is used to play songs entered with ENTRY. Songs can be read from cassette tape or from disk. Although songs cannot be edited with PLAY, it has several advantages over ENTRY. PLAY's main advantage is that it requires less memory than ENTRY. This means that PLAY can be loaded (from tape or disk) faster than ENTRY, and it allows playback of songs which are too large to load with ENTRY. Another important feature of PLAY is that most disk commands can be used (ENTRY allows only LOAD and SAVE). This allows "Exec Files" to be used, either as created by the DISCO program or custom files.

To run PLAY, you must have 5K bytes of memory plus enough additional memory to hold the song. If you are using a DISK II, you need 15.5K plus the song length. (Using the Applesoft verion, these figures are 8K and 18.5K.) The maximum song length is 28K. (17.5K for songs entered using a DISK II system with MAXFILES 3.)

First, load the program from disk or cassette tape. List line 1Ø. It will be 1Ø SLOT=4 : UNITS=1. Find the proper SLOT and UNITS values for your system using the table in the INSTALLATION section. Carefully retype the line changing only the digits 4 and 1 to the proper digits for your system. (If you have a Timing Mode Input Board, list line 2Ø. It will be 2Ø TSLOT=8. Carefully retype the line changing only the digit 8 to the slot number of your Input Board). Now save the program on your disk. If you do not have a DISK II, save the program using your own recorder to improve loadability. The program is now configured for your system, and can be run any time you like without having to change line 1Ø (or 2Ø). If you ever change the slot position of your synthesizer(s) (or Input Board), or purchase an additional synthesizer or an Input Board, you should do this configuration procedure again.

When run, PLAY will print a period (.) as a prompt character. The following commands can then be used:

**LOAD[:<song name>[<disk specifications>]]**
This command is the same as the load command in ENTRY (see the ENTRY section, SUMMARY OF COMMANDS).

**PLAY[:<song name>[<disk specifications>]]**
This command is a mixture of the play command in ENTRY (see the ENTRY section, SUMMARY OF COMMANDS) and the load command (above). Typing PLAY (return) is used to play the song currently in memory (you must have already loaded a song, of course). PLAY:<song name>[<disk specifications>] is used to load a song and then play it.

STOP
This command is used only in ALBUM files created by DISCO (see the DISCO

section). It goes to BASIC, leaving the PLAY program in memory for continuation with RUN. Either RUN or INT (FP when using Applesoft) should always be used after a STOP command.

**INT or FP**

INT (or FP for Applesoft) is used to stop using PLAY. The PLAY program is erased and must be reloaded if you desire to run it again.

Most disk commands, such as CATALOG and EXEC, can be used while running PLAY.

ENTRY's PLAY:F is not available in PLAY since the F would be assumed to be a song name.

If you wish to stop playback, press RESET. On systems not equipped with an Auto-Start ROM, type 3DØG (control C return on cassette systems) to return to BASIC. Once in BASIC, type RUN to clear the synthesizer and continue using PLAY.

# 5

# DISCO

The DISCO program is used to create an "Exec File" which can be used to play songs in succession. It can also randomize the playback order. It can be used only on systems equipped with a DISK II. A text file named ALBUM is created, so a disk which is not write-protected is required. The procedure is as follows:

Load DISCO from cassette tape or disk, and save it on your disk. (If you have already done this, just LOAD the program from your disk.) Type RUN 1ØØØ and press return. DISCO will print a brief set of instructions.

It is best if you have a printed catalog listing for this next step. If you don't have one, just type CATALOG occasionally to see the catalog listing. Type in the song names to be played, pressing return after each song name. Do not type the "M:" (for example, if you used SAVE:GALACTIC TRIUMPH from ENTRY, then you should type GALACTIC TRIUMPH (return) for DISCO, rather than M:GALACTIC TRIUMPH which is how the song will appear in the catalog). If you wish to have the songs played in a particular order, you must type them into DISCO in that order.

When all songs have been entered, type STOP and press return. **CAUTION:** care must be taken to not hold down the keyboard keys while typing STOP. The lack of n-key rollover on the Apple keyboard will cause unseen control letters to be entered if several keys are held down at once. This would cause a song title to be entered which consists of STOP and these control letters, rather than a STOP command.

If you wish to always use the same playback order, type LOCK ALBUM and press return. It will be necessary to type UNLOCK ALBUM if you ever wish to delete the ALBUM file or make any changes to it.

To play the whole sequence (or "album") of songs, you type EXEC ALBUM and press return. If you wish to have the order randomized, type RUN DISCO (or, if DISCO is already loaded, type RUN). To do either of these, a properly configured PLAY program must be on the disk and named PLAY. When album playback is complete, you can type RUN to run PLAY or EXEC ALBUM to hear the songs again. Otherwise, type INT or FP to stop using PLAY.

**TO ADD SONGS**

Load the DISCO program, and type RUN 2ØØØ (return). After the instructions are printed, proceed in the same fashion as when originally creating the album (done with RUN 1ØØØ, above).

**TO START OVER**

If you wish to scratch the old ALBUM file and make a new one, type DELETE ALBUM (return). Then LOAD DISCO and RUN 1ØØØ as described above. If you do not DELETE

ALBUM, and if the new ALBUM file is shorter than the old one, commands remaining at the end of the file will result in errors after album playback is completed.

### USING "START" and "END"

When randomizing the song order using RUN DISCO, you can have one particular song played as the first song, and/or another played as the last. These songs must be named START (for the first song) or END (for the last song). When a song named END is entered (during RUN 1ØØØ or RUN 2ØØØ), DISCO stops (there is no need to use a STOP command). END will remain the last song even if more songs are added (using RUN 2ØØØ) or the order is randomized (using RUN). The song must appear in the catalog as M:END. The START song should generally be entered as the first song, when the album is first made using RUN 1ØØØ. Otherwise it will not be the first song until it is randomly placed as the first (but will remain first from then on). It must appear in the catalog as M:START.

### USING MORE THAN ONE DISK DRIVE

Songs in an album can occupy more than one disk drive. The ALBUM file and the PLAY program must be on the same disk (as must be the START song, if used). (The END song, if used, must be on all disks.) Songs must be entered (when using RUN 1ØØØ or RUN 2ØØØ) followed by the proper disk specification. For example, when using two drives on the same controller, all songs on drive 1 must be followed by ",D1" and all songs on drive 2 must be followed by ",D2" (note: START and END must not be followed by a disk specification). If you are not using the randomization feature, the disk specifications need only be given when there is a change (for example, when the previous song was on drive 1 but this song is on drive 2, it must be followed by ",D2"). Be sure to leave enough room on the disk containing the ALBUM file for possible expansion of the file. **NOTE**: song titles are limited to 28 letters, including the disk specifications.

# 6

# PROGRAMMING

## WITH PERFORM

The PERFORM program is used to play songs from your own programs. It can play songs entered with ENTRY, or songs created by other means (see the SONG DATA description in this section).

PERFORM is rather difficult to use on systems which do not have a DISK II. In this case, PERFORM must be loaded from tape and RUN. PERFORM will then be located at 8Ø2 hex (2Ø5Ø decimal) in memory. LOMEM is automatically changed so PERFORM will not be erased by other programs you may load (note: be sure to avoid using control B or programs which change LOMEM). To use PERFORM, you must have a song in memory. At 8ØØ hex (2Ø48 decimal) you must put the starting address of the song MOD 256. (In Applesoft, this is address-INT(address/256)*256 since MOD is not available.) At 8Ø1 hex (2Ø49 decimal) you must put the starting address of the song divided by 256. (In Applesoft, this is INT(address/256).) Then, a CALL to 8Ø2 hex (2Ø5Ø decimal) causes the song to be played. The remainder of this section assumes you have a DISK II, but only the loading methods are different when using a cassette system (and the proper loading method has just been described). All explanations regarding the song data format are the same for any system. (**Note:** when using Applesoft, the word **LOMEM** in this paragraph refers to the start-of-program pointer.)

When using a system with a DISK II, you should change the PERFORM program into a binary file. Since you will probably want to still use the name PERFORM, you will have to delete the original PERFORM program since two programs cannot have the same name. To be on the safe side, you should begin by saving the original PERFORM program on some disk for possible future use. (Be sure you just LOAD PERFORM and then SAVE PERFORM on another disk. Do not RUN it or it will not be properly saved.) To begin, type INT (FP on Applesoft systems). Now load the PERFORM program (from cassette tape or disk). If you loaded it from disk, and wish to have the binary version of PERFORM on the same disk, you must DELETE PERFORM. Now RUN the program. Then type BSAVE PERFORM,A2Ø5Ø,L676 and press return. A binary file version of PERFORM will be saved on the disk. To finish, type INT (FP on Applesoft systems).

To copy the binary version of PERFORM to another disk, type BLOAD PERFORM,A2Ø5Ø to load it, and then BSAVE PERFORM,A2Ø5Ø,L676 to save it on the desired disk.

If you wish to play an ENTRY-created song from your own BASIC program, it will first be necessary to convert the song into a binary file so your program can load it. In order to play a song, its data must be initialized to have the correct SLOT and UNITS settings for your system. The easiest way to do this is to run a properly configured ENTRY program (see the ENTRY section), load the song and play it, then save the song back on disk. ENTRY's PLAY command will configure the song. (Note: you must remember to SAVE the configured song back on disk, or the disk copy of the song will not be configured.) Once you have done this, you are ready to convert the song into a binary file. (Note that it will be necessary to reconfigure the song if you change the slot location(s) of your synthesizers or add another synthesizer.) The following Integer BASIC program converts songs into binary files. Type it in and save it. Note that "d" means to type control D.

```
1Ø POKE 76,Ø : POKE 77,124 : DIM A$(4Ø) : INPUT "SONG NAME?",A$
2Ø PRINT "dLOADM:";A$ : A=PEEK(2Ø2)+PEEK(2Ø3)*256
3Ø PRINT "dBSAVE";A$;",A";A;",L";31744-A : PRINT "LENGTH: ";31744-A
4Ø PRINT "dINT"
```

Note: this program requires 48K. On 32K systems, change the 124 to an 8Ø and the two 31744's to 2Ø48Ø's. Songs entered on a 48K system with MAXFILES less than 3 (or on a cassette based system) may be too large to convert on a 32K system.

If you do not have Integer BASIC, use the following Applesoft version instead. Type it in and save it. Note that "d" means to type control D.

```
1Ø POKE 76,PEEK(115) : POKE 77,PEEK(116) : POKE 217,Ø
2Ø HIMEM:3ØØØ : INPUT "SONG NAME?";A$
3Ø POKE PEEK(54)+PEEK(55)*256+3Ø65,Ø
4Ø PRINT "dLOADM:";A$ : A=PEEK(2Ø2)+PEEK(2Ø3)*256
5Ø L=PEEK(76)+PEEK(77)*256-A : PRINT "dBSAVE";A$;",A";A;",L";L
6Ø PRINT "LENGTH: ";L : PRINT "dFP"
```

To use either program, begin by typing INT (FP for the Applesoft version). Then RUN the program. It will ask for a song name. Type in the name of the song to be converted (without the M:) and press return. The song will be converted and saved on your disk as a binary file with the same name as the song but without the M:. The conversion program also prints the length of the song in bytes. Although this length can be determined simply by BLOADing the song and looking at the DOS 3.2 file length locations (see your DOS manual), you may wish to write the length down since you will probably need to know it. To convert another song, follow the instructions above again. You can omit the initial INT (or FP), but you must load the program again to run it (or use RUN name) since the program self-destructs each time it is used.

## AN EXAMPLE

Let's say you want to try this procedure with the sample song MUSETTE. First, store a binary version of PERFORM as described above, and save the conversion program given above. Let's assume you named the conversion program CONVERT.

Now, RUN ENTRY. (This assumes you have already configured ENTRY for your system configuration as described in the ENTRY section.) LOAD:MUSETTE, PLAY, and SAVE:MUSETTE. Now type INT to exit ENTRY. You are now ready to convert the song. Type INT (or FP). Type RUN CONVERT. It will ask for a song name. Type MUSETTE and press return. The song will be converted and saved on your disk as MUSETTE, and the length will be printed. (If you had another song to convert now, you would start with RUN CONVERT.) Now, the song can be played with PERFORM. To do this, begin with BLOAD PERFORM. Now type BLOAD MUSETTE,A296Ø and then POKE 2Ø48,144 and POKE 2Ø49,11. Type CALL 2Ø5Ø to play the song. Note that paddle Ø controls the playback speed. When playback is finished, you could play the song again just by typing CALL 2Ø5Ø.

What are the mystic pokes for? Locations 2Ø48 and 2Ø49 must be set to the starting memory address of the song data. We loaded the song at 296Ø. Note that 11*256+144 (11 and 144 being the numbers we poked) is 296Ø, the starting address. 296Ø just happens to be the first byte of memory available after PERFORM, which uses locations 2Ø48 through 2959.

With a few precautions, you could have had a BASIC program do the BLOADs, POKEs, and CALL. The only other detail is that in this example we used ENTRY to initialize the synthesizers (when MUSETTE was configured), and for general purpose BASIC programs you would probably want to have your program initialize the synthesizers. If you were writing a program to be used on other people's computers, you would probably want to have your program configure the song data, too.

## A FEW PRECAUTIONS

When using PERFORM from a BASIC program, you will have to find a place to put the song data. You will also have to keep BASIC from erasing PERFORM.

### WITH INTEGER BASIC

When using Integer BASIC, the easiest place to put the song data is right after PERFORM. (Starting at 296Ø decimal.) LOMEM can be moved up to keep BASIC from erasing either PERFORM or the song data. First, figure out where the song data will end. You will need to know the length of the longest song you plan to BLOAD, or the sum of the lengths of the longest songs you plan to have in memory at the same time. Take this length and add the starting address (296Ø). This is what LOMEM must be. You can either set LOMEM using a LOMEM command, or you can have your program set LOMEM. It is probably best to have your program do it so you won't forget, and so others can use it. The LOMEM command also changes a value Apple calls CM, so your program must change it too. To do all this, find out what LOMEM MOD 256 and LOMEM/256 are (for the new LOMEM, of course). For example, if your longest song is less than 2Ø48 bytes, LOMEM could be 296Ø+2Ø48 which is 5ØØ8. 5ØØ8 MOD 256 is 144 and 5ØØ8/256 is 19. To have your program change LOMEM and CM to these values, make the first statements POKE 74,144 : POKE 2Ø4,144 : POKE 75,19 : POKE 2Ø5,19. These four pokes must be the first statements in your program, or at least be before any variables are used. After these, you can BLOAD PERFORM by using PRINT "dBLOAD PERFORM" where the "d" is a control D. You can load a song using PRINT "dBLOAD song name,A296Ø" where "song name" is the name of the song to be loaded. If you wish to load a second song, change the 296Ø after the A to a value which is 296Ø plus the length of the first song or greater. Similarly, a third song can be loaded with an A value of 296Ø plus the combined lengths of all previously loaded songs (or greater). Loading several songs lets you do a lot of disk reading at the beginning of the program (or any time before playback is needed) and then play any of the loaded songs at any time without delay. On the other hand, you may wish to just load a song, play it, then load another song and play it. This requires less memory, and it splits up the disk reading time. (When reading one song at a time, you only need enough memory to hold the longest song, and you BLOAD each song at the same address.) To play any song, you will need its starting address. This is the number after the A in the BLOAD command. The address MOD 256 must be poked at 2Ø48, and the address/256 must be poked at 2Ø49. Then a CALL 2Ø5Ø is used to play the song. If you load another song at the same address, you don't need to poke the starting address again. However, if you've loaded several songs, you will need to poke the starting address of the desired song before using CALL 2Ø5Ø to play it.

You can, of course, locate your song data any place it won't be erased. You must still move LOMEM up to at least 296Ø to keep Integer BASIC from erasing PERFORM.

**WITH APPLESOFT BASIC**

Applesoft's memory organization is very crude, and thus more awkward preparations (than with Integer BASIC) are required. To begin with, your program should start with several REM statements. They should be line numbers 1 through 4. Line 1 should be typed in as 1REMXXXX... with no spaces and with enough X's to completely fill 6 lines on the Apple's 4Ø column display. (There will be 235 X's. Don't type the ... of course.) Lines 2 through 4 must start as 2REMXXXX... 3REMXXXX... etc. These REM statements provide enough room for the PERFORM program. The next lines in your program should change Applesoft's start-of-program pointer to eliminate the REM's (while keeping enough room available for PERFORM). This is done with the statements POKE 1Ø3,197 : POKE 1Ø4,11. This is all you need to keep Applesoft from erasing PERFORM. Now an area of memory for the song(s) to be played is needed. The easiest area to use is the memory below the DOS system (below HIMEM). You will need to know the length of the longest song you plan to BLOAD, or the sum of the lengths of the longest songs you plan to have in memory at the same time. Let's call this number "length". Before your program uses any variables, you should have this statement to reserve an area of memory (of length "length") for the song(s): IF PEEK(2Ø5Ø)<>138 THEN HIMEM:PEEK(115)+PEEK(116)*256-length. After this, you can use PRINT "dBLOAD PERFORM" to read in the PERFORM program (remember that "d" means to type control D). Now you should set a variable which indicates the address of this memory area. This is done with the statement A=PEEK(115)+PEEK(116)*256. You can load a song using PRINT "dBLOAD song name,A";A where "song name" is the name of the song to be loaded. If you wish to load a second song, use PRINT "dBLOAD song name,A";A+L where "song name" is the name of the second song, and L is the length of the first song. Similarly, a third song can be loaded using a value for L which is the combined lengths of all previously loaded songs. Loading several songs lets you do a lot of disk reading at the beginning of the program (or any time before playback is needed) and then play any of the loaded songs at any time without delay. On the other hand, you may wish to just load a song, play it, then load another song and play it. This requires less memory, and it splits up the disk reading time. (When reading one song at a time, you only need enough memory to hold the longest song, and you BLOAD each song at the same address.) To play any song, you will need to poke its starting address at 2Ø48 and 2Ø49. The starting address is the value A (or A+L) in the BLOAD command. Use POKE 2Ø48,A-INT(A/256)*256 : POKE 2Ø49,A/256. Then a CALL 2Ø5Ø is used to play the song. If you load another song at the same address, you don't need to poke the starting address again. However, if you've loaded several songs, you will need to poke the starting address of the desired song before using CALL 2Ø5Ø to play it.

You can, of course, locate your song data any place it won't be erased. You must still use the REM statements and the POKE 1Ø3,197 : POKE 1Ø4,11 to keep Applesoft from erasing PERFORM.

**CAUTION:** when you run your Applesoft program, the REM statements will disappear. This will present no problems unless you save the program while the REM statements are gone. If you do, then sometime later (when PERFORM is no longer in memory) you may run the program and the first few lines would disappear, possibly causing bizzare listings (due to partial lines) and really odd RUNs after the first one. To repair this problem, just load the missing REM version from the disk and type in the REMs. To avoid having this problem occur, begin any session of correction by loading the program, running it to make the REMs disappear, then loading it again to bring the REMs back; this time the REMs will not disappear when you run the program since the start-of-program pointer has already been changed.

## SYNTHESIZER INITIALIZATION

If you have a line which sets SLOT and UNITS, like the one in ENTRY or PLAY, you can use these variables in a synthesizer initialization routine. Generally, any program which uses the synthesizer should have this initialization routine near the beginning. It is the same for either Integer BASIC or Applesoft.

```
FOR S=SLOT TO SLOT+UNITS-1
PN=16*S-16256 : POKE PN,Ø : POKE PN+1,Ø : POKE PN+2,Ø
POKE PN+3,3 : POKE PN+7,54 : POKE PN+7,118 : POKE PN+7,182
NEXT S
```

## SONG CONFIGURATION

Unless you can configure each song for your particular system (using ENTRY, as previously described) and can count on your program being used only on your system, you will need a song configuration routine. This routine uses SLOT and UNITS, as does the synthesizer initialization routine (above). It also needs the variable A set to the starting address of the song to be configured.

The Integer BASIC version looks like this: (note: the song must not occupy address 32768)

```
FOR B=1 TO PEEK(A)
PNTR=PEEK(B+B+A-1)+PEEK(B+B+A)*256+A : CHAN=B-1
IF UNITS>1 THEN CHAN=PEEK(PNTR+2)/(1+15*(3-UNITS))
CHAN=CHAN MOD I6 : POKE PNTR+I,CHAN/4*12+CHAN+SLOT*16
NEXT B
```

The Applesoft version looks like this:

```
FOR B=1 TO PEEK(A)
PNTR=PEEK(B+B+A-1)+PEEK(B+B+A)*256+A : CHAN=B-1
IF UNITS>1 THEN CHAN=PEEK(PNTR+2)/(1+15*(3-UNITS))
CHAN=CHAN-INT(CHAN/16)*16 : POKE PNTR+1,INT(CHAN/4)*12+CHAN+SLOT*16
NEXT B
```

When using either version, you might wish to add POKE 2Ø48,A MOD 256 : POKE 2Ø49,A/256 : CALL 2Ø5Ø : RETURN (which is POKE 2Ø48,A-INT(A/256)*256 : POKE 2Ø49,A/256 : CALL 2Ø5Ø : RETURN in Applesoft) to the end in order to create a subroutine which can be GOSUBed in order to configure and play the song at address A.

## READING THE "SUGGESTED SPEED"

Assuming the song was just loaded using PRINT "dBLOAD song name,A";A the suggested speed from an ENTRY-created song can be read into the variable S with the following statement: S=PEEK(PEEK(-2192Ø)+PEEK(-21919)*256+A-161). Note that the entire song must be located below memory address 32768 when using Integer BASIC. The -2192Ø is for a 48K system and must be -383Ø4 on a 32K system. Likewise, the -21919 must be -383Ø3 on a 32K system. In either case, Apple's DOS 3.2 must be used.

## TEMPO CONTROL

If you wish to use a different paddle than Paddle Ø to control the playback speed, you must POKE 2345,n where n is the paddle number plus 1ØØ. (For a fixed playback speed, you may wish to install a 15ØK ohm 1/4 watt resistor at the game paddle connector between the +5 and PDL3 pins; then select paddle 3 for playback control as just described. Paddle 3 is an ideal choice since there is no switch input for this paddle, which may prohibit use of a real paddle. For additional information, request application note AN8Ø-1.)

The following routine modifies PERFORM for timing mode and initializes channel Ø of the proper synthesizer for timing mode operation. TSLOT must be set to the slot number of the timing mode input board, or to 8 when the game I/O input is used, as it is for ENTRY and PLAY.

```
S=(SLOT+(UNITS>1))*16+132 : POKE 2113,S : POKE 2118,S
POKE 2345,99+(TSLOT*16+29)*(TSLOT<8) : POKE 2347,16
POKE S-16388,Ø : POKE S-16381,48
```

To go to normal mode, use: POKE 2113,32 : POKE 2118,112 : POKE 2345,1ØØ : POKE 2347,48 : POKE (SLOT+(UNITS>1))*16-16249,54. Note that the POKE 2345,1ØØ should be 1ØØ plus the paddle number. The last poke (with SLOT and UNITS) is needed only if the synthesizer is not going to be initialized prior to its next use.

## A SAMPLE SESSION

The following sample session is for a 48K system with Integer BASIC and Apple's DOS 3.2. The changes necessary for a 32K system or for Applesoft (or both) have already been discussed above. It is assumed that the PERFORM program and the M:MUSETTE song, as provided with the synthesizer, are already on disk.

```
>LOAD PERFORM
>DELETE PERFORM
>RUN
PERFORM   ALF PRODUCTS INC.

>BSAVE PERFORM,A2Ø5Ø,L676
```

**CONVERT PERFORM TO A BINARY FILE**

```
>INT
>1Ø POKE 76,Ø : POKE 77,I24 : DIM A$(4Ø) : INPUT "SONG NAME?",A$
>2Ø PRINT "dLOADM:";A$ : A=PEEK(2Ø2)+PEEK(2Ø3)*256                  SAVE
>3Ø PRINT "dBSAVE";A$;",A";A;",L";31744-A : PRINT "LENGTH: ";31744-A  CONVERT
>4Ø PRINT "dINT"                                                      PROGRAM
>SAVE CONVERT
>RUN
SONG NAME?MUSETTE                                                     CONVERT
                                                                      MUSETTE
LENGTH: 1146                                                          TO BINARY

>5 POKE 74,Ø : POKE 2Ø4,Ø : POKE 75,64 : POKE 2Ø5,64  PROTECT PERFORM
>1Ø SLOT=4 : UNITS=1   (CHANGE AS REOUIRED)          AND SONG AREA
>2Ø TSLOT=8
>3Ø PRINT "dBLOAD PERFORM" : DIM A$(4Ø)   LOAD PERFORM
>4Ø FOR S=SLOT TO SLOT+UNITS-1    INITIALIZE SYNTHESIZER
>5Ø PN=16*S-16256 : POKE PN,Ø : POKE PN+1,Ø : POKE PN+2,Ø
>6Ø POKE PN+3,3 : POKE PN+7,54 : POKE PN+7,118 : POKE PN+7,182
>7Ø NEXT S
>8Ø INPUT "SONG NAME?",A$ : A=296Ø : PRINT "dBLOAD";A$;",A";A   READ SONG
>9Ø S=PEEK(PEEK(-2192Ø)+PEEK(-21919)*256+A-161)   READ SUGGESTED SPEED
>1ØØ IF S THEN 14Ø
>11Ø S=(SLOT+(UNITS>1))*16+132 : POKE 2113,S : POKE 2118,S   TIMING MODE
>12Ø POKE 2345,99+(TSLOT*16+29)*(TSLOT<8) : POKE 2347,16
>13Ø POKE S-I6388,Ø : POKE S-1638I,48 : GOTO 18Ø
>14Ø POKE 2113,32 : POKE 21I8,112 : POKE 2345,1ØØ   NORMAL MODE
>15Ø POKE 2347,48 : POKE (SLOT+(UNITS>1))*16-16249,54
>16Ø PRINT "SUGGESTED SPEED: ";S
>17Ø PRINT PDL(Ø);"  "; : TAB I : IF PEEK(-16287)<128 THEN I7Ø
>18Ø FOR B=I TO PEEK(A)          EITHER MODE: CONFIGURE SONG
>19Ø PNTR=PEEK(B+B+A-1)+PEEK(B+B+A)*256+A : CHAN=B-I
>2ØØ IF UNITS>1 THEN CHAN=PEEK(PNTR+2)/(1+15*(3-UNITS))
>21Ø CHAN=CHAN MOD 16 : POKE PNTR+1,CHAN/4*12+CHAN+SLOT*16
>22Ø NEXT B : POKE 2Ø48,A MOD 256 : POKE 2Ø49,A/256
>23Ø CALL 2Ø5Ø : GOTO 8Ø     PLAY THE SONG
>SAVE YALP
>RUN
SONG NAME?MUSETTE

SUGGESTED SPEED: I9Ø
(etc.)          (SONG PLAYS WHEN BUTTON IS PRESSED)
```

**CREATION OF A SIMPLE PROGRAM**

## TECHNICAL

PERFORM operates on one to nine sequences of commands stored in memory. Each sequence of commands indicates the sounds for one channel on one synthesizer. All the sequences will appear to be executed at the same time by PERFORM. There are three types of commands which may be used. One type is used to control the execution of the commands. Another type is used to set parameters for future use. The remaining type of command is used to wait or to produce a new pitch and wait. During the time "waited", PERFORM will automatically program volume settings which create the selected envelopes. Envelope production is explained in the ENTRY section and in the block diagram at the end of this section.

All commands for PERFORM are three bytes long. (Each byte is an integer from Ø to 255.) The first byte always indicates the particular command desired, and the second and third bytes indicate a parameter for use by that command. When the parameter is a two-byte integer (Ø to 65535), the low byte (value MOD 256) is given as the second byte of the command and the high byte (value/256) is given as the third byte. The various commands available are described below.

## TYPE A COMMANDS

The first type of command is used to control execution. They are CHANNEL NUMBER, CALL, RETURN, STOP, and END.

### CHANNEL NUMBER

The CHANNEL NUMBER command is used to indicate the slot and channel number to be programmed. The second byte should be 16 times the expansion slot number plus the channel number. Although PERFORM does not use the third byte, it should be used to indicate stereo positioning. Its most significant four bits indicate stereo positioning for performance with two units (meaningless in songs that have more than six parts), and the least significant four bits indicate stereo positioning for performance with three units. In each half byte, the two most significant bits indicate the relative unit number (Ø to 2). This number can be added to SLOT to create the actual unit number. The two least significant bits indicate the channel number (Ø to 2). Thus, the second byte must be computed by multiplying the "actual unit number" (above) by 16 and adding the "channel number". The first command in each part must be a CHANNEL NUMBER command. ENTRY compatible songs may have only one CHANNEL NUMBER command per part.

### CALL

The CALL command is used to perform a subroutine call. The second and third bytes indicate the relative address of the subroutine. During playback, the commands in the subroutine will be executed, and then PERFORM will continue in the usual fashion with the commands following the CALL.

### RETURN

The RETURN command marks the end of a subroutine, and causes PERFORM to continue at the commands following the CALL. The second and third bytes must be the same as the second and third bytes of the CALL command. ENTRY compatible songs may have only one RETURN command per subroutine.

### STOP

The STOP command indicates the end of one part's (or channel's) commands. The envelope generator will continue to operate after a STOP command if no other channel has encountered an END command. The second and third bytes are not used and should be set to Ø. All parts except the last one should end with a STOP command. ENTRY compatible songs may have only one STOP command per part.

### END

The END command is used to terminate PERFORM and return to the calling program. The last part should end with an END command rather than a STOP command. Further, the END command should be positioned as the last command in all the data (in ENTRY compatible songs, this is followed by the "suggested speed" byte and the 16Ø title bytes). Envelope production does not continue once any part executes an END command. The second and third bytes are not used, and should be set to Ø.

## TYPE B COMMANDS

The second type of command is used to set parameters. They are TRANSPOSE, GAP SIZE, ATTACK RATE, DECAY RATE, VOLUME LEVEL, SUSTAIN LEVEL, and RELEASE RATE.

### TRANSPOSE

The TRANSPOSE command is used to add or subtract a constant from all following pitches (until a new TRANSPOSE value is programmed). The second byte indicates the amount to add or subtract. Ø to 127 will add a value of Ø to 127. 128 to 255 will subtract a value of 128 to 1. Since the values are in quarter-steps, adding a value of 24 will raise the pitch by one octave. The third byte is the pitch mask byte. All following pitch values are ANDed with the pitch mask byte (before the second byte transpose value is added or subtracted). This byte is normally set to 255. ENTRY compatible songs use a value of 254 to allow sharp/flat display selection with the least significant pitch bit.

### GAP SIZE

The GAP SIZE command is used to control the release stage of envelope production. When the number of time periods remaining to wait (during a "wait") equals the GAP SIZE value, the envelope parameters will automatically be changed. The RELEASE RATE value will be copied into the CURRENT DECAY RATE, and a Ø will be written into the DESIRED LOUDNESS and the CURRENT SUSTAIN LEVEL. This causes the CURRENT LOUDNESS (and therefore the volume) of the channel to drop to Ø at the RELEASE RATE. The second and third bytes indicate the new GAP SIZE. When a release stage is not desired, the GAP SIZE should be set to 65535 (255,255).

### ATTACK RATE, DECAY RATE, VOLUME LEVEL, SUSTAIN LEVEL, RELEASE RATE

These commands are used to set envelope parameters. The second and third bytes indicate the new value.

## TYPE C COMMANDS

The third type of command is used to wait or to produce a new pitch and wait. The second and third bytes indicate the number of time periods to wait before continuing with the next command. During this wait, the envelope generator program in PERFORM will update the envelope parameters and reprogram the volume once each time period. These commands are PITCH and REST.

### PITCH

There are 192 PITCH commands with command numbers from Ø to 191. The command number indicates which pitch is to be produced, subject to modification by the two TRANSPOSE parameters. The resultant number specifies the pitch to be programmed into the synthesizer. Pitch specification is in quarter-steps, with Ø being A natural at 27.5 Hz. There are 24 quarter-steps per octave. Thus, 24 is A natural at 55 Hz. Note that in ENTRY compatible songs, the least significant bit of the PITCH command number indicates whether sharp or flat should be displayed, and is masked off during playback (see TRANPOSE). The PITCH command also changes certain envelope parameters. The DECAY RATE is copied into the CURRENT DECAY RATE, the VOLUME LEVEL is copied into the DESIRED LOUDNESS, and the SUSTAIN LEVEL is copied into the CURRENT SUSTAIN LEVEL (see the block diagram at the end of this section).

### REST

The REST command causes the RELEASE RATE to be copied into the CURRENT DECAY RATE, and a Ø to be written into the DESIRED LOUDNESS and the CURRENT SUSTAIN LEVEL. This causes the release portion of the envelope to begin. (Note: this is the same process as caused by the time remaining equaling the GAP SIZE, see the GAP SIZE command.)

## SONG DATA

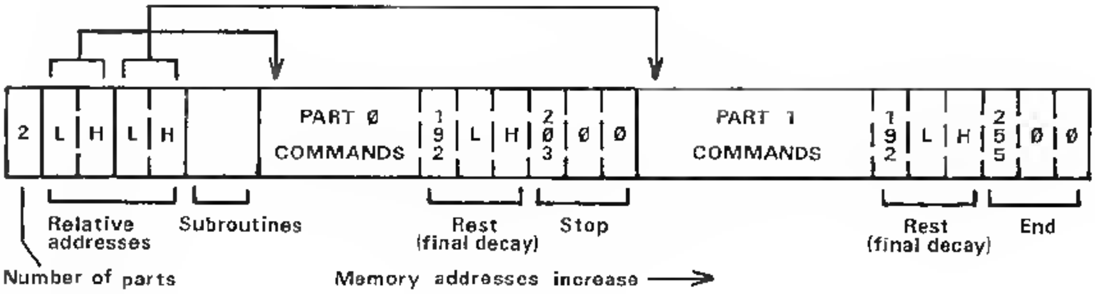
### RELATIVE ADDRESSES

All relative addresses used in PERFORM (for example, the second and third bytes of a CALL command) must be two-byte integers stored low byte first. The value stored must be the actual memory address minus the starting address of the song data.

### START OF OATA

The first byte (stored at the starting address) must be the number of "parts" of data. This must be an integer from 1 to 9. The following 2 to 18 bytes must be the relative address of the first command of each part. Following these bytes the subroutines (if any) are stored, and then the first part's commands, the second's, and so forth. See the diagram below.

Two Part Song Data



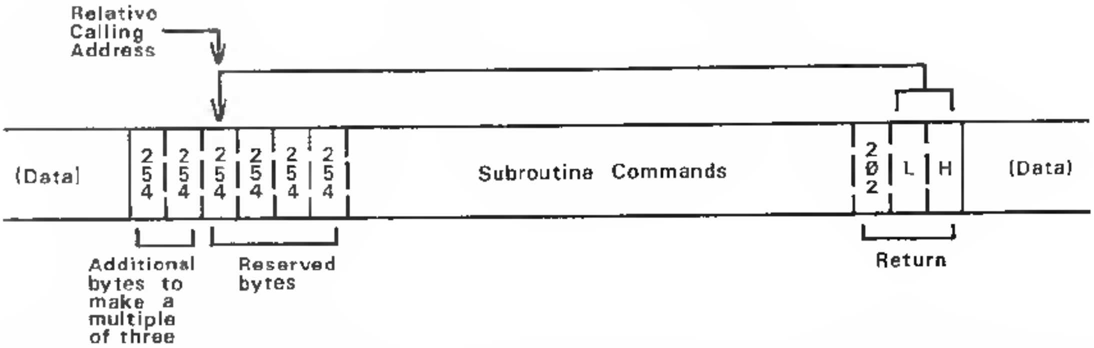
### PART DATA

In each part, the three-byte commands are stored one after another. Each part must begin with a CHANNEL NUMBER command, and end with a STOP command (except the last part must end with an END command). See the diagram above. Although a part may contain more than one CHANNEL command, to do so would be incompatible with ENTRY and with the "song configuration" routine given earlier in this section.

### SUBROUTINE OATA

The relative calling address to a subroutine must point to several bytes of reserved storage which preceed the first command of the subroutine. There must be two times as many reserved bytes as the number of parts. These reserved bytes must be preceeded by at least 1 additional byte(s), and the number of additional bytes plus the number of reserved bytes must be evenly divisible by 3. See the diagram below.

Subroutine (in two part song data)



Note that the calling address must point to the first of the reserved bytes, not to the additional bytes nor to the first command in the subroutine. The additional bytes must be stored as 254's, and the reserved bytes should be set to 254 also. When a CALL command is executed during playback, the address of the first command after the CALL (that is, the return address) is stored in two of the reserved bytes. (PERFORM assigns a different pair of bytes for each part. This allows several parts to call the subroutine at once.) The RETURN command at the end of the subroutine causes the address of the next-command-to-be-interpreted to be read from the correct pair of reserved bytes, thus causing a "return". Note that although a subroutine may contain more than one RETURN command (or a RETURN command to a different subroutine), to do so would be incompatible with ENTRY.

## TEMPO COMMAND

The TEMPO command is a rather unusual command. It is used to dynamically control playback tempo (speed). At the start of each time period, a two byte value is written to a selected synthesizer's channel Ø (only when using Timing Mode). This channel must have been previously initialized to Timing Mode. This two byte value determines the length of a time period, which will be value/1782ØØØ seconds. The second and third bytes of the TEMPO command indicate a new value. Since the Timing Mode synthesizer channel controls the playback speed for all parts, the TEMPO command can appear in any part. Note that when using Timing Mode, channel Ø of one synthesizer (the higher numbered slot when using two synthesizers, or the middle slot when using three) cannot be used to play music. Its volume should be programmed to Ø.

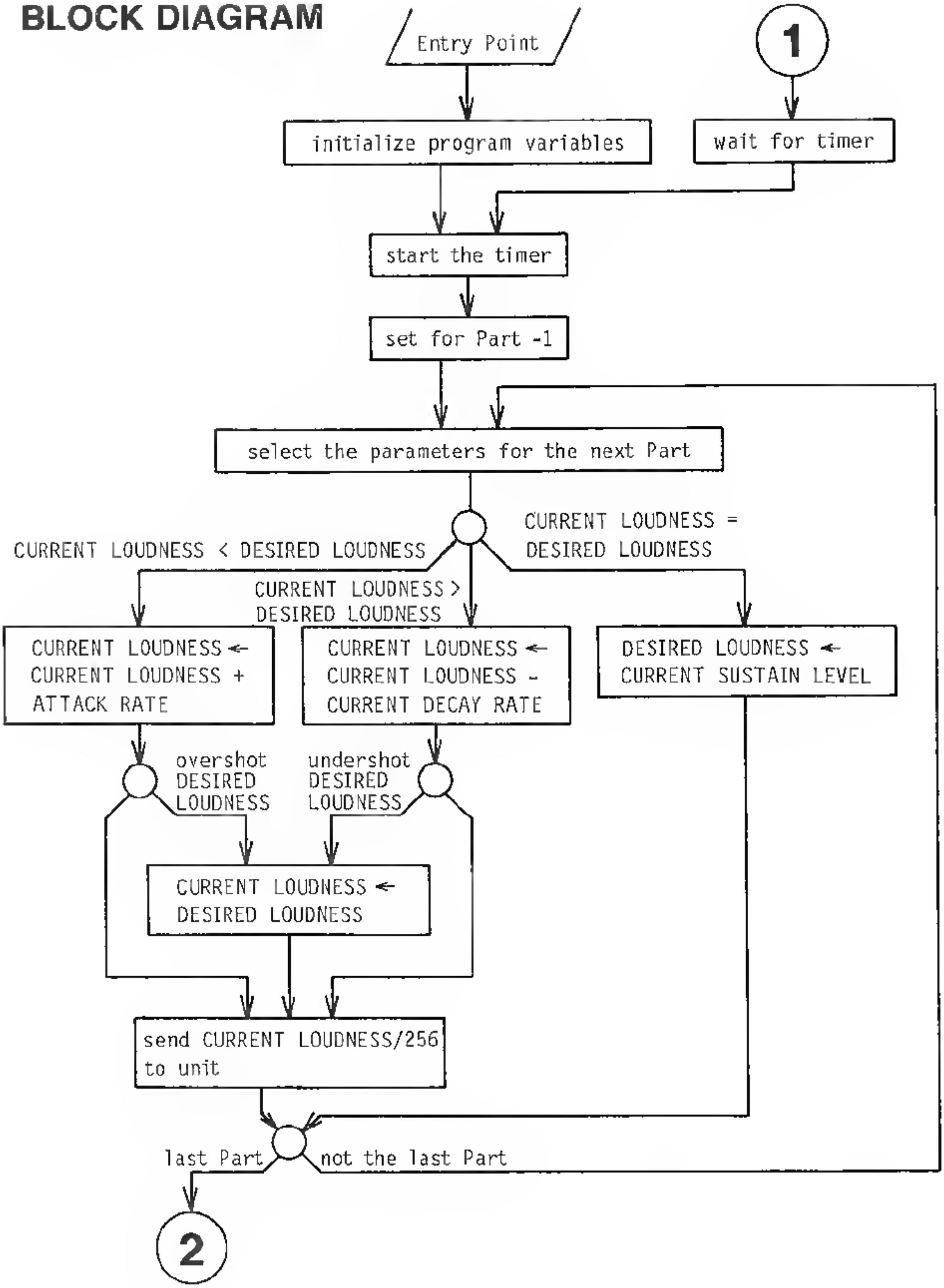
## TEMPORARIES

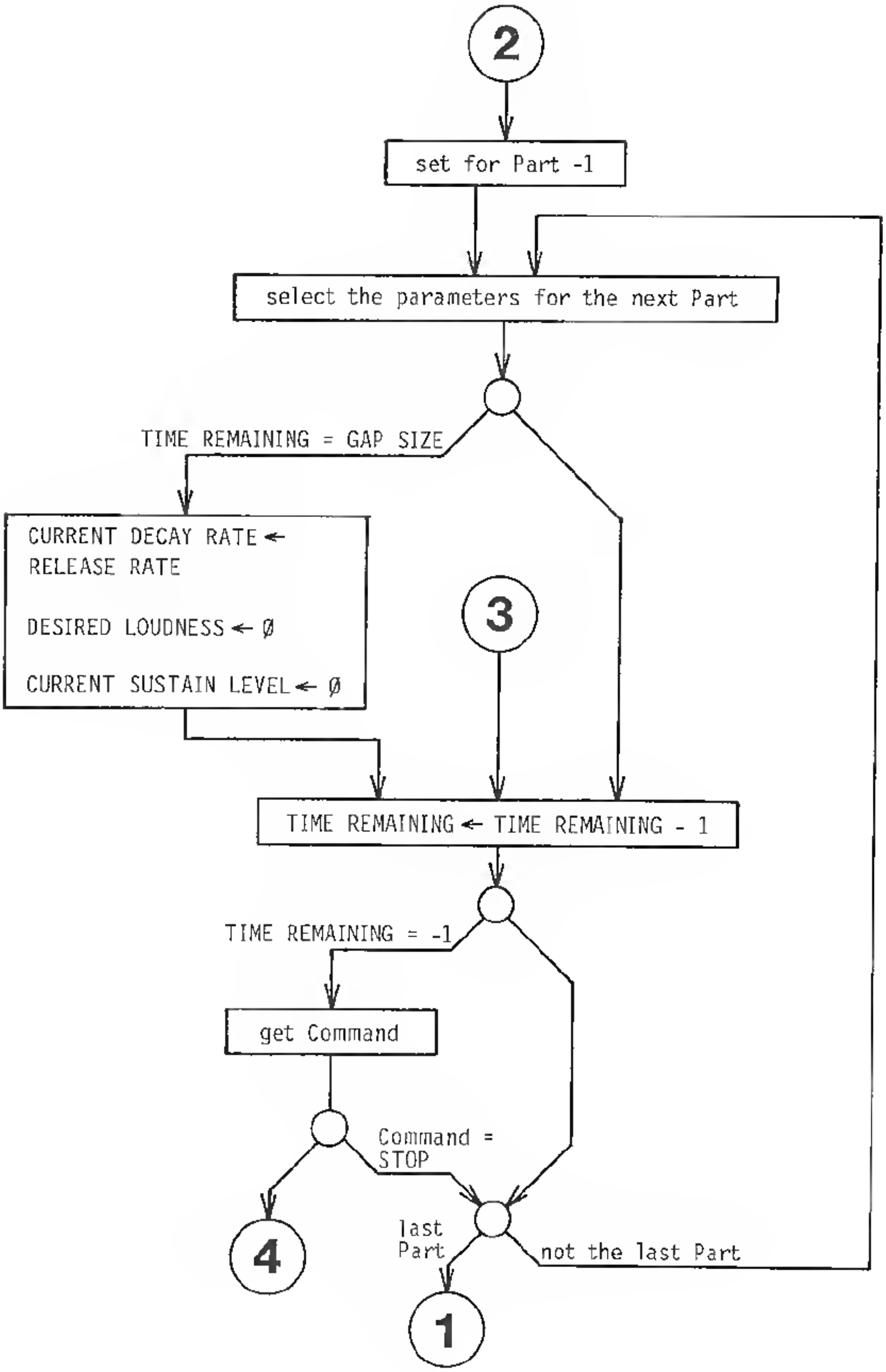
PERFORM uses locations Ø-19 (hex) (6-C and DD-EF for the Applesoft version) for storage of temporary values during execution.

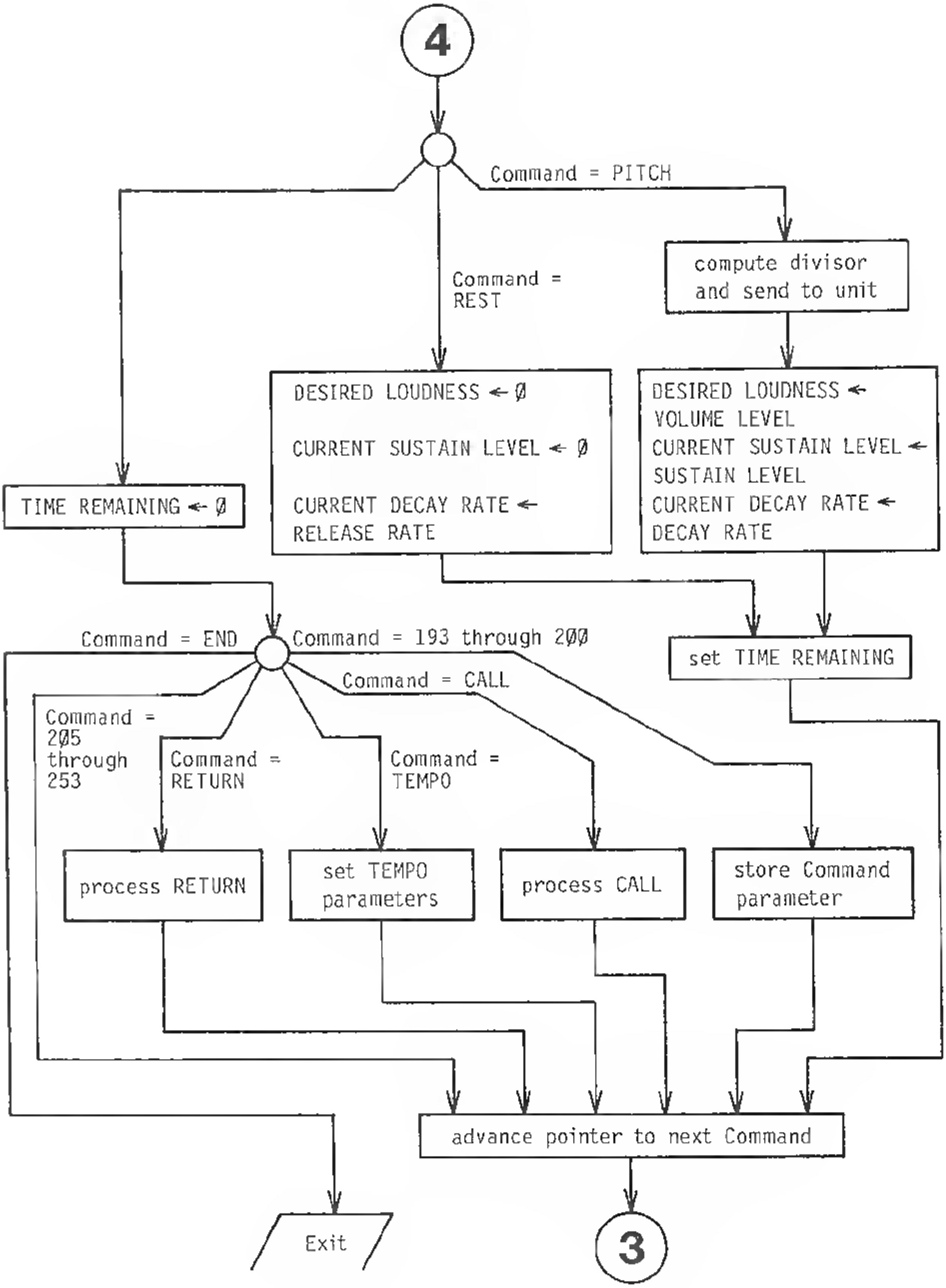
## COMMAND NUMBERS

| HEX | DECIMAL | COMMAND |
|---|---|---|
| Ø-BF | Ø-191 | PITCH |
| CØ | 192 | REST |
| C1 | 193 | GAP SIZE |
| C2 | 194 | TRANSPOSE |
| C3 | 195 | ATTACK RATE |
| C4 | 196 | DECAY RATE |
| C5 | 197 | VOLUME LEVEL |
| C6 | 198 | SUSTAIN LEVEL |
| C7 | 199 | RELEASE RATE |
| C8 | 2ØØ | CHANNEL NUMBER |
| C9 | 2Ø1 | CALL |
| CA | 2Ø2 | RETURN |
| CB | 2Ø3 | STOP |
| CC | 2Ø4 | TEMPO |
| CD-FD | 2Ø5-253 | no operation |
| FE | 254 | preceeds subroutines, treated as END if found |
| FF | 255 | END |

## BLOCK DIAGRAM


Entry Point
1
initialize program variables
wait for timer
start the timer
set for Part -1
select the parameters for the next Part
CURRENT LOUDNESS < DESIRED LOUDNESS
CURRENT LOUDNESS = DESIRED LOUDNESS
CURRENT LOUDNESS > DESIRED LOUDNESS
CURRENT LOUDNESS ← CURRENT LOUDNESS + ATTACK RATE
CURRENT LOUDNESS ← CURRENT LOUDNESS - CURRENT DECAY RATE
DESIRED LOUDNESS ← CURRENT SUSTAIN LEVEL
overshot DESIRED LOUDNESS
undershot DESIRED LOUDNESS
CURRENT LOUDNESS ← DESIRED LOUDNESS
send CURRENT LOUDNESS/256 to unit
last Part
not the last Part
2


2
set for Part -1
select the parameters for the next Part
TIME REMAINING = GAP SIZE
CURRENT DECAY RATE ← RELEASE RATE
DESIRED LOUDNESS ← Ø
CURRENT SUSTAIN LEVEL ← Ø
3
TIME REMAINING ← TIME REMAINING - 1
TIME REMAINING = -1
get Command
Command = STOP
4
last Part
not the last Part
1


4
Command = PITCH
Command = REST
compute divisor and send to unit
TIME REMAINING ← Ø
DESIRED LOUDNESS ← Ø
CURRENT SUSTAIN LEVEL ← Ø
CURRENT DECAY RATE ← RELEASE RATE
DESIRED LOUDNESS ← VOLUME LEVEL
CURRENT SUSTAIN LEVEL ← SUSTAIN LEVEL
CURRENT DECAY RATE ← DECAY RATE
set TIME REMAINING
Command = END
Command = 193 through 2ØØ
Command = CALL
Command = 2Ø5 through 253
Command = RETURN
Command = TEMPO
process RETURN
set TEMPO parameters
process CALL
store Command parameter
advance pointer to next Command
Exit
3

# 7

# PROGRAMMING

## WITH CHROMA

The CHROMA subroutine is used to simplify programming the synthesizer with chromatic (equal tempered) pitches. The various routines in CHROMA are:

1. **INITIALIZER.** Written in BASIC, this routine initializes the synthesizer, the CHROMA routine, and the PULSE routine.
2. **PARTIAL INITIALIZER.** Written in BASIC, this routine is used to initialize additional synthesizers.
3. **CHROMA.** Written in 65Ø2 assembly language, this routine is used to program "normal mode" (square wave) pitches.
4. **PULSE.** Written in 65Ø2 assembly language, this routine is used to program "pulse mode" (pulse wave) pitches.

The parameters required by these routines, their calling procedures, functions, and results are described below.

## INITIALIZER

The INITIALIZER uses the value of the variable SLOT. Prior to calling the INITIALIZER, this variable should be set to the expansion slot number one of your synthesizers is plugged into. The INITIALIZER is called using GOSUB 32767. It will initialize the synthesizer, correct memory addresses in the CHROMA and PULSE routines, assign values to the variables PITCH and VOLØ, and poke SLOT*16 at PITCH+2 and Ø at PITCH+3 (see table below). "Initialize the synthesizer" means to set all three channels to zero volume and "normal mode".

| POKE ADDRESS | NAME | OESCRIPTION |
|---|---|---|
| PITCH | PITCH | Pitch number |
| PITCH+1 | PART | Channel (part) number |
| PITCH+2 | | Slot number times 16 |
| PITCH+3 | OFFSET | Pitch offset |
| PITCH+4 | WIDTH | Pulse width |
| PITCH+5 | | Divisor low |
| PITCH+6 | | Divisor high |
| PITCH+7 | CHROMA | CHROMA entry point |
| (PITCH+8 and PITCH+9 are reserved.) | | |
| PITCH+1Ø | PULSE | PULSE entry point |
| (PITCH+11 and PITCH+12 are reserved.) | | |
| PITCH+13 | | (start of divisor table) |

The table above shows the memory locations used for parameter storage by the CHROMA and PULSE routines. The address of this table is indicated by the value assigned to PITCH, which is based on the value of HIMEM (or the length of your program when using Applesoft). Note that when using Integer BASIC, HIMEM must not be -32498, -32433, or any value in between.

The variable VOLØ is used to set volume levels and change modes.

| **POKE ADDRESS** | **NAME** | **DESCRIPTION** |
|---|---|---|
| VOLØ | VOLØ | Volume for channel Ø |
| VOLØ+1 | VOL1 | Volume for channel 1 |
| VOLØ+2 | VOL2 | Volume for channel 2 |
| VOLØ+3 | | Mode control A |
| VOLØ+7 | | Mode control B |

Values poked at the above addresses go directly to the synthesizer and cause the volume or mode to change immediately. Values from Ø to 255 can be poked for volume (Ø=off or 1=soft to 255=loud). The following values can be poked for mode control (other values should not be used).

| **POKE ADDRESS** | **VALUE** | **FUNCTION** |
|---|---|---|
| VOLØ+3 | Ø | Both channels Ø and 1 to pulse mode |
| VOLØ+3 | 1 | Channel Ø to normal mode, channel 1 to pulse mode |
| VOLØ+3 | 2 | Channel Ø to pulse mode, channel 1 to normal mode |
| VOLØ+3 | 3 | Both channels Ø and 1 to normal mode |
| VOLØ+7 | 5Ø | Channel Ø to pulse mode |
| VOLØ+7 | 54 | Channel Ø to normal mode |
| VOLØ+7 | 114 | Channel 1 to pulse mode |
| VOLØ+7 | 118 | Channel 1 to normal mode |
| VOLØ+7 | 182 | Channel 2 to normal mode (used by the INITIALIZER) |

The INITIALIZER and PARTIAL INITIALIZER set all three channels to normal mode. To change modes, set the mode by poking the value shown above to VOLØ+7, then the appropriate value (above) to VOLØ+3.

The value assigned to VOLØ by the INITIALIZER or PARTIAL INITIALIZER is different for each expansion slot and is calculated by the formula VOLØ=SLOT*16-16256.

The mnemonic variable names shown in the first table can be set using the following statements. (Note: the variable name PART was given as CHANNEL, which is more appropriate, in previous manuals. However, Applesoft does not allow two variables to be named CHANNEL and CHROMA.) The setup and calling of the INITIALIZER is included:

```
1Ø SLOT=4      (replace 4 with the proper slot number)
2Ø GOSUB 32767 : PART=PITCH+1 : OFFSET=PITCH+3 : WIDTH=PITCH+4 :
   CHROMA=PITCH+7 : PULSE=PITCH+1Ø : VOL1=VOLØ+1 : VOL2=VOLØ+2
```

**NOTE:** Applesoft does not allow three variables to be named VOLØ, VOL1, and VOL2. Applesoft users should pick names for VOL1 and VOL2 (if they need these variables) which do not begin with the same 2 letters as any other variable, and complain to Microsoft.

## PARTIAL INITIALIZER

When more than one synthesizer is used, the units not initialized with the INITIALIZER (GOSUB 32767) must be initialized with the PARTIAL INITIALIZER. For each additional board, set SLOT to the proper expansion slot number, and call the PARTIAL INITIALIZER using GOSUB -2. It will initialize the synthesizer and set VOLØ to the volume control address for that slot number. Previous values of VOLØ set by the INITIALIZER or PARTIAL INITIALIZER should be assigned to other variables if they must be retained. (The value of VOLØ for any slot is computed by the formula VOLØ=SLOT*16-16256.) Note that GOSUB -2 does not cause the slot number times 16 to be written at PITCH+2 or a zero to be written at PITCH+3. GOSUB -3 can be used instead if you wish to have these values poked. (On systems where Applesoft doesn't allow GOSUB with negative numbers, use 63998 instead of -2 and 63997 instead of -3.)

## CHROMA

CHROMA uses the parameters poked at PITCH, PART, PITCH+2, and OFFSET. It changes the contents of PITCH+5 and PITCH+6. When called using CALL CHROMA (or CALL PITCH+7), CHROMA programs the desired channel (indicated by PART) on the desired synthesis board (indicated by the slot number times 16 at PITCH+2) with the desired pitch (indicated by PITCH and OFFSET). To do this, CHROMA will calculate a two-byte divisor which it stores at PITCH+5 and PITCH+6 in case it is needed for PULSE (see the PULSE routine in this section). The precise function of these poked parameters is as follows:

**PART (PITCH+1)**

This indicates which of the three channels is to be programmed. It must be an integer from Ø to 2. Adding 128 will inhibit programming of the synthesizer but the divisor will still be computed and stored.

**PITCH+2**

This indicates the slot number of the synthesizer to be programmed. The value poked must be the slot number (Ø to 7) times 16. If only one synthesizer is used, this parameter does not need to be poked since it is initialized to SLOT*16 by the INITIALIZER.

**PITCH**

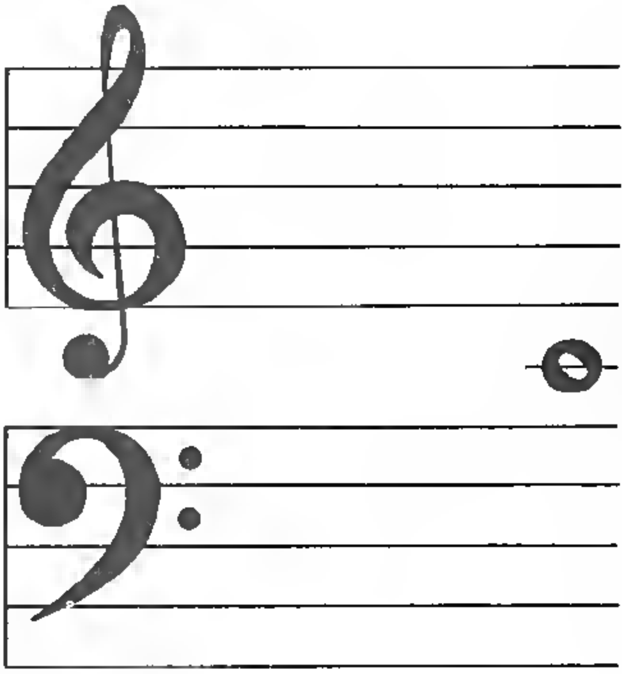
This indicates the quarter-tone pitch to be programmed. The values for half-tones in the lowest octave are:

| | | | | | |
|---|---|---|---|---|---|
| Ø | A | 8 | C sharp | 16 | F |
| 2 | A sharp | 1Ø | D | 18 | F sharp |
| 4 | B | 12 | D sharp | 2Ø | G |
| 6 | C | 14 | E | 22 | G sharp |

For quarter-tones, add 1. For higher octaves, add the numbers shown below to the numbers shown above. The frequency of the A in that octave is also shown below. (Note: "octaves" here start at A.)

| A (Hz) | Add | A (Hz) | Add | A (Hz) | Add | A (Hz) | Add |
|---|---|---|---|---|---|---|---|
| 27.5 | Ø | 11Ø | 48 | 44Ø | 96 | 176Ø | 144 |
| 55 | 24 | 22Ø | 72 | 88Ø | 12Ø | 352Ø | 168 |

The highest pitch (G sharp plus a quarter-step) in the highest octave is 22+1+168 (or 191), so pitch values should be from Ø to 191. Some common notes and their values are (for sharp, add 2; for flat, subtract 2):



| Hex | Decimal | Note |
|---|---|---|
| 7Ø | 112 | F |
| 6E | 11Ø | E |
| 6A | 1Ø6 | D |
| 66 | 1Ø2 | C |
| 64 | 1ØØ | B |
| 6Ø | 96 | A 44Ø |
| 5C | 92 | G |
| 58 | 88 | F |
| 56 | 86 | E |
| 52 | 82 | D |
| 4E | 78 | Middle C |
| 4C | 76 | B |
| 48 | 72 | A |
| 44 | 68 | G |
| 4Ø | 64 | F |
| 3E | 62 | E |
| 3A | 58 | D |
| 36 | 54 | C |
| 34 | 52 | B |
| 3Ø | 48 | A |
| 2C | 44 | G |

### OFFSET (PITCH+3)

This indicates how sharp the pitch should be from standard tuning. Ø is used for standard A=44Ø Hz tuning (as initialized by GOSUB 32767 or GOSUB -3), and numbers from 1 to 255 are used to raise the pitch slightly. All pitches selected using OFFSET are less than or equal to the pitch selected by a PITCH setting one higher. Note that the pitches selected by various values of PITCH increase exponentially, whereas the pitches selected by various values of OFFSET (with a constant PITCH setting) increase linearly.

# PULSE

The PULSE routine is used to create pulse waves using channel Ø and/or channel 1. The frequency (pitch) of the pulse wave will be the same as the frequency of channel 2. The INITIALIZER sets all channels to normal mode, so channels to be used with PULSE must be changed to "pulse mode" as previously described. The parameters poked at PART, PITCH+2, WIDTH, PITCH+5, and at PITCH+6 are used. PULSE is called using CALL PULSE (or CALL PITCH+1Ø). The precise function of each parameter is as follows:

**PART (PITCH+1)**

This indicates which of the two channels is to be programmed. It must be either Ø or 1. Adding 128 will inhibit programming of the synthesizer but the divisor will still be calculated and stored (see divisor storage locations below).

**PITCH+2**

This indicates the slot number of the synthesizer to be programmed. The value must be the slot number (Ø to 7) times 16.

**WIDTH (PITCH+4)**

This indicates the width of the low part of each cycle. Numbers from Ø to 126 indicate a short low portion, and numbers from 128 to 255 indicate a long low portion. I27 is used to program a square waveform.

**PITCH+5 and PITCH+6**

These must contain the divisor currently programmed for channel 2. If CHROMA was called most recently for channel 2, these locations will already be set to the divisor (by CHROMA).

The divisor calculated by PULSE is stored at locations 81 and 82 decimal (6I and 62 in Applesoft). It may be read using peek immediately after calling PULSE.

## CHROMA EXAMPLE

To program a three note chord of Middle C, E, G at maximum volume, begin by loading CHROMA. Now type in the following program, remembering to change the 4 to the correct expansion slot number.

```
1Ø SLOT=4
2Ø GOSUB 32767 : PART=PITCH+1 : OFFSET=PITCH+3 : CHROMA=PITCH+7
3Ø POKE PART,Ø : POKE PITCH,78 : CALL CHROMA : POKE VOLØ,255
4Ø POKE PART,1 : POKE PITCH,86 : CALL CHROMA : POKE VOLØ+1,255
5Ø POKE PART,2 : POKE PITCH,92 : CALL CHROMA : POKE VOLØ+2,255
6Ø END
```

Now run the program. The synthesizer will be programmed for the C E G chord, and it will continue to produce the chord until programmed to do something else. The chord can be cleared by typing GOTO -2.

## PULSE EXAMPLE

The following program produces one tone with the pitch controlled by Paddle Ø and the pulse width controlled by Paddle 1. As in the above example, begin by loading CHROMA. Then add the program below, remembering to correct the slot number.

```
1Ø SLOT=4
2Ø GOSUB 32767 : PART=PITCH+1 : WIDTH=PITCH+4
3Ø CHROMA=PITCH+7 : PULSE=PITCH+1Ø : POKE VOLØ+7,5Ø : POKE VOLØ+3,2
4Ø POKE PART,2 : POKE PITCH,PDL(Ø)/2 : CALL CHROMA
5Ø POKE PART,Ø : POKE WIDTH,PDL(1) : CALL PULSE
6Ø POKE VOLØ,255 : GOTO 4Ø
```

Now run the program, and twist the paddle knobs like mad. Stop the program, and type POKE VOLØ,Ø to stop the noise.

# 8

# PROGRAMMING

## BARE HANDED

The Apple Music Synthesizer is programmed by means of 8 "ports". Each port is assigned a particular memory address, and information can be sent to a port by writing a byte (an integer from Ø to 255) to that memory address (using 65Ø2 Assembly Language or BASIC's POKE). Reading from these memory addresses does not affect the synthesizer. The ports are numbered from Ø to 7. The memory address of each port is calculated by the formula SLOT*16-16256+P where SLOT is the expansion slot number used by the synthesizer and P is the desired port number (both should be Ø to 7).

The function of each port is as follows:

| **PORT** | **FUNCTION** |
|---|---|
| Ø | Volume control for channel Ø |
| 1 | Volume control for channel 1 |
| 2 | Volume control for channel 2 |
| 3 | Mode control A |
| 4 | Divisor for channel Ø |
| 5 | Divisor for channel 1 |
| 6 | Divisor for channel 2 |
| 7 | Mode control B |

**Ports Ø-2** are used to control the volume. A byte written to one of these ports will cause the volume of the appropriate channel to change immediately to the new value (Ø=off or 1=soft to 255=loud). The relative output voltage for any volume setting (VOL) is computed by 2↑(VOL/32)*(VOL MOD 32 + 33)-33 with Integer BASIC, or by 2↑INT(VOL/32)*(VOL-INT(VOL/32)*32 + 33)-33 with Applesoft BASIC.

**Ports 3 and 7** are used for mode control. Before use, all channels must be initialized to either normal mode or pulse mode to insure proper operation. Port 3 selects whether the pitch control will be provided by the Apple or by the output of Channel 2. Port 7 selects whether the divisor will control the pitch or the pulse width. Normally both ports 3 and 7 are set to indicate either normal mode or pulse mode. Port 7 should be programmed before port 3 for best results.

The value written to port 3 has the following effects:

| **VALUE** | **MEANING** |
|---|---|
| Ø | Both channels Ø and 1 to pulse mode |
| 1 | Channel Ø to normal mode, channel 1 to pulse mode |
| 2 | Channel Ø to pulse mode, channel 1 to normal mode |
| 3 | Both channels Ø and 1 to normal mode |

Other values should not be used.

Values written to port 7 have the following effects:

| VALUE | MEANING |
|---|---|
| 5Ø | Set channel Ø to pulse mode, channels 1 and 2 not affected |
| 54 | Set channel Ø to normal mode, channels I and 2 not affected |
| 114 | Set channel 1 to pulse mode, channels Ø and 2 not affected |
| 118 | Set channel 1 to normal mode, channels Ø and 2 not affected |
| 182 | Set channel 2 to normal mode, channels Ø and 1 not affected |

Other values should not be used except as noted in the TIMING MODE section.

When a channel is set to a mode using port 7, the output of its pitch generator will go high and stay high until both bytes of a divisor are written. The high part of the cycle will then begin. (Note: port 3 should be set after port 7 is set but before the first divisor is programmed.)

When a channel is set to pulse mode with port 7 but normal mode with port 3, the output of its pitch generator will stay high. When a channel is set to pulse mode with port 3 but normal mode with port 7, the output of its pitch generator will be high when the output of channel 2's pitch generator is low, and when the channel 2 output goes high the mixed-mode channel will begin normal square wave operation starting with the high part of the cycle. (Once the channel 2 output returns to low, the mixed-mode channel will go high and stay high until the channel 2 output goes high again.)

Any of the three channels can also be set to a special "timing mode" where the channel is used to simulate the Apple "paddle" timers, but with a programmable setting. See the TIMING MODE section for details.

**Ports 4-6** are used to program the divisor. Once a channel has been initialized, it will be expecting the low byte of the divisor (D MOD 256). Once the low byte is written, it will be expecting the high byte of the divisor (D/256). Once the high byte is written, the new divisor will be used by the pitch generator; and the low byte of the next divisor will be expected.

When in normal mode, the divisor determines the frequency to be produced by the pitch generator. The duty cycle is always approximately 5Ø% and cannot be altered. The output frequency will be I782ØØØ/D Hz (where D is the divisor programmed) plus or minus Ø.Ø15%. The value D must be an integer from 32 to 65536. (Note: 65536 must be programmed as Ø. Values less than 32 are possible but should not be used.) When a new divisor is programmed, it does not take effect until the associated pitch generator's output goes high after the high byte of the divisor was written.

When in pulse mode, the divisor determines the time duration of the low portion of the pulse wave. The frequency is determined by the frequency output of channel 2's pitch generator. Just after the low to high change of channel 2's pitch generator output, the output of the pulse mode channel's pitch generator will go low. It will stay low for D/1782ØØØ seconds plus or minus Ø.Ø15%. If the channel 2 output has again gone high during this time, the pulse mode output will stay low. Otherwise, the pulse mode output will go high and stay high until the next time the channel 2 output goes high. The value D must be an integer from 1 to 65536. (Note: 65536 must be programmed as Ø.) When a new divisor is programmed, it does not take effect until the first low to high change in the output of channel 2's pitch generator after the high byte of the divisor was written.

## DIVISOR CALCULATION

Pitches and volumes must increase (and decrease) exponentially to achieve an apparent linear increase (for humans). Exponential volume increases are automatically created by the exponential amplifiers in the volume control circuitry. Exponential pitch increases must be created by selecting divisors which result in exponentially higher (and lower) pitches.

The most common exponential pitch spacing is the equal tempered scale, which is similar to the piano scale. This scale is divided into "octaves" with 12 notes per octave (half tones) or 24 notes per octave (quarter tones) depending on the application. An octave is defined to mean that the frequency of a note is twice that of the same note in the next lower octave. The frequency, F(N), of any particular note, N, in an octave is calculated by F(N)=F(Ø)*(2↑(N/X)) where X is the number of notes per octave, F(Ø) is the frequency (pitch) of the lowest note in the octave (in Hz, or cycles per second), and N must be an integer from Ø to X-1. (Note: although written in standard BASIC format, the formulas here are not intended to be computed in BASIC without careful consideration of the accuracy required. Floating-point calculation should be used in any case.) The frequency, F(N,Q), of any given note, N, in any given octave, Q, is calculated F(N,Q)=F(N,Ø)*(2↑Q) where F(N,Ø) is equivalent to F(N) in the previous formula and Q is an integer. The lowest note on a piano has a frequency of 27.5 Hz (using standard A=44Ø Hz tuning). Thus the frequency, F(N,Q), of any piano note is F(N,Q)=27.5*(2↑(Q+N/12)) Hz, where N is the note number from Ø to 11 and Q is the octave number from Ø to 7. (Note: pianos have no notes where N is greater than 3 if Q is 7. N=Ø indicates an A natural pitch.) Therefore the desired divisors for piano notes are: D(N,Q)=INT(1782ØØØ/(27.5*(2↑(Q+N/12)))+Ø.5). Note that the 12 can be replaced with a 24 (and the range of N extended to Ø-23) to obtain quarter tones. It is usually convenient to calculate divisors using a small look-up table containing D(N,Ø) and dividing by 2↑Q and rounding. This is easily accomplished in assembly language by shifting the divisor right Q times (shifting in Ø's) and then adding in the last bit shifted out in order to round.

## TUNING

It may be useful to know that musicians use "cents" to express the amount of deviation from correct tuning for half tones. A note too high (sharp) by 1ØØ cents would be the right frequency for the next higher half step. The formula for cents is (12ØØ*LOG(F/X))/LOG(2) where X is the correct frequency in Hz and F is the actual frequency produced in Hz. (The LOG may be in any base, as long as it is always the same base.) Inaccurate tuning in the synthesizer's pitches results mainly from the fact that only integral values can be used for the divisor (D). This creates pitches out of tune by amounts varying from Ø to Ø.Ø2Ø cents in the lowest 12 notes of the piano scale, which increase to Ø.Ø67 to 1.2Ø4 cents in the top 12 notes. (The Ø.Ø15% crystal accuracy adds a maximum of Ø.26Ø cents.) Tuning accuracy within 2 cents should be considered excellent and suitable for any purpose.

# 9

# TIMING MODE

When playing songs with PERFORM (see the PERFORM section), song tempo (playback speed) is normally controlled by the setting of paddle Ø. (Note: since both ENTRY and PLAY use PERFORM, this section applies to playback with ENTRY and PLAY as well as PERFORM.) The paddles on the Apple actually control hardware timers, which (when using the PDL functions) the software measures the time delay of in order to produce a number from Ø to 255. PERFORM uses this time delay to control the playback speed directly, so the physical positioning of the paddle knob (not the imaginary Ø to 255 number) adjusts the speed. In many applications, this may be undesirable. It is especially undesirable in two particuarly common procedures. One is the use of DISCO for continuous playback of songs. Songs generally have a variety of paddle settings, and it is inconvenient to have to re-adjust the paddle knob position between each song. The second occurs in songs which have ritards or similar tempo changes from one section to another. It would be inconvenient to create such changes by manually adjusting the knob while the song plays.

Fortunately, the TEMPO command can be used to select any of a variety of playback speeds. (See the PERFORM and ENTRY sections.) However, the TEMPO command is only used when "timing mode" is activated. Timing mode is a special mode in which one channel of one synthesizer is programmed to function similar to the Apple paddle timers. The pitch programmed into that channel determines the delay time (and thus the playback speed) rather than a physical knob position. Naturally, this means that one synthesizer channel cannot be used for normal playback, since it is occupied with the timing tone.

The software provided with the synthesizer is only programmed for timing mode using channel Ø of a particular synthesizer. (The higher numbered slot when using 2 synthesizers, and the middle slot when using 3.) When writing your own software, any channel can be used.

## CONNECTION

In order to use timing mode, the output of the channel to be used must be connected into the Apple's hardware so its status can be read. There are two simple ways to do this. The easiest method is to use the Timing Mode Input Board (ALF part number 1Ø-5-17) which plugs into any expansion slot in the Apple, and connects to the empty socket on a synthesizer. However, if it is undesirable to use an additional slot, a channel output can be connected to the Apple Game I/O connector using a simple "header to header" cable (ALF part number 1Ø-1-8), and the Game I/O Socket Extender (ALF part number 1Ø-1-9) which allows both the game paddles and the header to header cable to be plugged in at the same time. Using either scheme, the cables are constructed to use channel Ø (as required by standard ALF software). Those who wish to make a header to header cable themselves should connect pin 3 of a 14-pin DIP IC header (for the empty socket on the synthesizer) to pin 4 (switch input 2) of a 16-pin DIP IC header (for the Apple Game I/O socket).

## ENTRY & PLAY

ENTRY and PLAY contain a line 2Ø which is normally 2Ø TSLOT=8. The 8 value selects the Game I/O connection method. Values from Ø to 7 select the Timing Mode Input Board connection method and also indicate which slot the TMIB is in. In either case, the header which plugs into the empty socket on the synthesizer must be connected to the higher numbered unit (the "right" unit) when using 2 synthesizers, or the middle unit when using 3.

It is important to note that the channel used for timing mode should not be assigned a part of the music. Thus, the number of parts which can be played when timing mode is activated (suggested speed=Ø) is 2, 5, or 8 (for 1, 2, or 3 synthesizers). When using the STEREO command (see the ENTRY section), you must remember that only 2 R's can be used if you have 2 units; or that only 2 M's can be used if you have 3 units. Since the assumed stereo for three units is MLRMLRMLR (which would have 3 M's if 7 or 8 parts are used), this must be changed after each EDIT command if the number of parts is changed to 7 or 8.

Remember that each song must begin with a TEMPO command in one of the parts before the first note or rest (or CALL to a subroutine with a note or rest). Traditionally, this is done in Part Ø.

## TECHNICAL

To initialize a channel to "timing mode", port 3 (mode control A) is set to "normal mode". The following value is sent to port 7 (mode control B): 48 for channel Ø, I12 for channel 1, or 176 for channel 2. Note that no mode control A setting is required for channel 2. The pitch generator output of the selected channel will go low upon initialization. Volume for the timing mode channel should be set to Ø unless you wish to hear the timing tone.

To "set" the timer, a two byte divisor, D, is sent in the normal fashion (see the BARE HANDED programming section). The output will go low (or stay low if it is already low). After D/1782ØØØ plus or minus Ø.Ø15% seconds, the output will go high, and stay high until the next divisor is programmed. This is the same as the Apple paddle timers, except the signal is inverted (the Apple timers go high when set and go low upon time-out).

# 10

# LISTINGS

# PERFORM (INTEGER VERSION)

```
ØØØØ            1Ø    * PERFORM SUBROUTINE
ØØØØ            2Ø    *
ØØØØ            3Ø    * BY JOHN RIDGES
ØØØØ            4Ø    *
ØØØØ            5Ø    * ALF PRODUCTS INC.
ØØØØ            6Ø    *
ØØØØ            7Ø              ORG Ø
ØØØØ            8Ø    * BASE PAGE USAGE
ØØØØ            9Ø    SPNTR     BSS 2                SONG DATA POINTER
ØØØ2            1ØØ   COUNT     BSS 1                PART COUNTER
ØØØ3            11Ø   TEMP1     BSS 1
ØØØ4            12Ø   TEMP2     BSS 1
ØØØ5            13Ø   TEMP3     BSS 2
ØØØ7            14Ø   PARNUM    BSS 19               NUMBER OF PARTS
ØØØ8            15Ø   PARPNT    EQU PARNUM+1         PART POINTERS
Ø8ØØ            I6Ø             ORG $8ØØ
Ø8ØØ            17Ø   * SUBROUTINE PARAMETER
Ø8ØØ            18Ø   DPNTR     BSS 2                SONG DATA BEGINNING ADDRESS
Ø8Ø2            19Ø   * SUBROUTINE ENTRY POINT
Ø8Ø2  8A        2ØØ             TXA                  SAVE X
Ø8Ø3  48        21Ø             PHA                   REGISTER
Ø8Ø4  AD ØØ Ø8  22Ø             LDA DPNTR            SET SONG
Ø8Ø7  85 ØØ     23Ø             STA /SPNTR            DATA POINTER
Ø8Ø9  AD Ø1 Ø8  24Ø             LDA DPNTR+1
Ø8ØC  85 Ø1     25Ø             STA /SPNTR+1
Ø8ØE  AØ ØØ     26Ø             LDY #Ø               GET NUMBER
Ø81Ø  B1 ØØ     27Ø             LDA (SPNTR),Y         OF PARTS
Ø812  ØA        28Ø             ASL A
Ø813  85 Ø7     29Ø             STA /PARNUM
Ø815  A2 ØØ     3ØØ             LDX #Ø               SET UP
Ø817  C8        31Ø   CPYADR    INY                   PART POINTERS
Ø818  B1 ØØ     32Ø             LDA (SPNTR),Y
Ø81A  I8        33Ø             CLC
Ø81B  65 ØØ     34Ø             ADC /SPNTR
Ø81D  95 Ø8     35Ø             STA /PARPNT,X
Ø81F  E8        36Ø             INX
Ø82Ø  C8        37Ø             INY
Ø821  B1 ØØ     38Ø             LDA (SPNTR),Y
Ø823  65 Ø1     39Ø             ADC /SPNTR+1
Ø825  95 Ø8     4ØØ             STA /PARPNT,X
Ø827  E8        41Ø             INX
Ø828  E4 Ø7     42Ø             CPX /PARNUM
Ø82A  DØ EB     43Ø             BNE CPYADR
Ø82C  46 Ø7     44Ø             LSR /PARNUM
Ø82E  A2 EA     45Ø             LDX #234             CLEAR
Ø83Ø  A9 ØØ     46Ø             LDA #Ø                PARAMETER AREA
Ø832  9D A5 ØA  47Ø   CLEAR     STA TIME-1,X
Ø835  CA        48Ø             DEX
Ø836  DØ FA     49Ø             BNE CLEAR
Ø838            5ØØ   * MAIN EXECUTION LOOP
Ø838  A5 Ø7     51Ø             LDA /PARNUM          SET UP
Ø83A  85 Ø2     52Ø             STA /COUNT            PART COUNTER
Ø83C  A2 ØØ     53Ø   MAIN      LDX #Ø
Ø83E  A9 ØØ     54Ø   PLACE1    LDA #Ø               RESERVE SPACE
Ø84Ø  8D 2Ø CØ  55Ø             STA $CØ2Ø             FOR TEMPO
Ø843  A9 ØØ     56Ø   PLACE2    LDA #Ø                 COMMAND
Ø845  8D 7Ø CØ  57Ø             STA $CØ7Ø            START TIMER
Ø848            58Ø   * ENVELOPE PROCESSING SECTION
```

```
Ø848  BD B8 ØA  59Ø   ENVEL    LDA LOUDNS,X       CHECK CL (CURRENT LOUDNESS)
Ø84B  38        6ØØ            SEC                 AND DL (DESIRED LOUDNESS)
Ø84C  FD BC ØA  61Ø            SBC DESIRE,X
Ø84F  85 Ø3     62Ø            STA /TEMP1
Ø851  BD B9 ØA  63Ø            LDA LOUDNS+1,X
Ø854  FD BD ØA  64Ø            SBC DESIRE+1,X
Ø857  9Ø 12     65Ø            BCC UPLD           BRANCH IF CL<DL
Ø859  Ø5 Ø3     66Ø            ORA /TEMP1
Ø85B  DØ 31     67Ø            BNE DWNLD          BRANCH IF CL>DL
Ø85D  BD BE ØA  68Ø            LDA CURSUS,X       CL=DL
Ø86Ø  9D BC ØA  69Ø            STA DESIRE,X       DL:=CURRENT SUSTAIN LEVEL
Ø863  BD BF ØA  7ØØ            LDA CURSUS+1,X
Ø866  9D BD ØA  71Ø            STA DESIRE+1,X
Ø869  BØ 66     72Ø            BCS NEXTE
Ø86B  BD B8 ØA  73Ø   UPLD     LDA LOUDNS,X       CL:=CL+ATTACK RATE
Ø86E  7D AC ØA  74Ø            ADC ATTACK,X
Ø871  9D B8 ØA  75Ø            STA LOUDNS,X
Ø874  BD B9 ØA  76Ø            LDA LOUDNS+1,X
Ø877  7D AD ØA  77Ø            ADC ATTACK+1,X
Ø87A  9D B9 ØA  78Ø            STA LOUDNS+1,X
Ø87D  BØ 31     79Ø            BCS ETHERE         BRANCH IF OVERSHOT DL
Ø87F  A8        8ØØ            TAY                COMPARE CL AND DL
Ø88Ø  BD B8 ØA  81Ø            LDA LOUDNS,X
Ø883  DD BC ØA  82Ø            CMP DESIRE,X
Ø886  98        83Ø            TYA
Ø887  FD BD ØA  84Ø            SBC DESIRE+1,X
Ø88A  9Ø 3C     85Ø            BCC SENDE          DON'T BRANCH IF OVERSHOT DL
Ø88C  BØ 22     86Ø            BCS ETHERE
Ø88E  BD B8 ØA  87Ø   DWNLD    LDA LOUDNS,X       CL:=CL-CURRENT DECAY RATE
Ø891  FD BA ØA  88Ø            SBC DOWN,X
Ø894  9D B8 ØA  89Ø            STA LOUDNS,X
Ø897  BD B9 ØA  9ØØ            LDA LOUDNS+1,X
Ø89A  FD BB ØA  91Ø            SBC DOWN+1,X
Ø89D  9D B9 ØA  92Ø            STA LOUDNS+1,X
Ø8AØ  9Ø ØE     93Ø            BCC ETHERE         BRANCH IF UNDERSHOT DL
Ø8A2  BD BC ØA  94Ø            LDA DESIRE,X       COMPARE CL AND DL
Ø8A5  DD B8 ØA  95Ø            CMP LOUDNS,X
Ø8A8  BD BD ØA  96Ø            LDA DESIRE+1,X
Ø8AB  FD B9 ØA  97Ø            SBC LOUDNS+1,X
Ø8AE  9Ø 18     98Ø            BCC SENDE          DON'T BRANCH IF UNDERSHOT DL
Ø8BØ  BD BC ØA  99Ø   ETHERE   LDA DESIRE,X       CL:=DL
Ø8B3  9D B8 ØA  1ØØØ           STA LOUDNS,X
Ø8B6  BD BD ØA  1Ø1Ø           LDA DES1RE+I,X
Ø8B9  9D B9 ØA  1Ø2Ø           STA LOUDNS+1,X
Ø8BC  BD BE ØA  1Ø3Ø           LDA CURSUS,X       DL:=CURRENT SUSTAIN LEVEL
Ø8BF  9D BC ØA  1Ø4Ø           STA DESIRE,X
Ø8C2  BD BF ØA  1Ø5Ø           LDA CURSUS+1,X
Ø8C5  9D BD ØA  1Ø6Ø           STA DESIRE+I,X
Ø8C8  BC B6 ØA  1Ø7Ø  SENDE    LDY CHAN,X         SEND LOUDNESS
Ø8CB  BD B9 ØA  1Ø8Ø           LDA LOUDNS+I,X      TO UNIT
Ø8CE  99 8Ø CØ  1Ø9Ø           STA $CØ8Ø,Y
Ø8D1  8A        11ØØ  NEXTE    TXA                REPEAT FOR
Ø8D2  18        111Ø           CLC                 NEXT PART
Ø8D3  69 1A     112Ø           ADC #ASIZE
Ø8D5  AA        113Ø           TAX
Ø8D6  C6 Ø2     114Ø           DEC /COUNT
Ø8D8  FØ Ø3     115Ø           BEQ CONT1
Ø8DA  4C 48 Ø8  116Ø           JMP ENVEL
Ø8DD  A2 ØØ     117Ø  CONT1    LDX #Ø             INITIALIZE PART COUNTER
Ø8DF            118Ø  * NOTE DURATION SECTION
Ø8DF  BD A6 ØA  119Ø  LENGTH   LDA TIME,X         COMPARE TIME REMAINING
```

```
Ø8E2  DD A8 ØA  12ØØ          CMP GAP,X          AND GAP SIZE
Ø8E5  DØ 22     121Ø          BNE DECR          BRANCH IF UNEQUAL
Ø8E7  BD A7 ØA  122Ø          LDA TIME+1,X
Ø8EA  DD A9 ØA  123Ø          CMP GAP+1,X
Ø8ED  DØ 1A     124Ø          BNE DECR          BRANCH IF UNEQUAL
Ø8EF  BD B4 ØA  125Ø          LDA RELEAS,X      EQUAL; START NOTE
Ø8F2  9D BA ØA  126Ø          STA DOWN,X         RELEASE
Ø8F5  BD B5 ØA  127Ø          LDA RELEAS+1,X      CURRENT DECAY RATE:=
Ø8F8  9D BB ØA  128Ø          STA DOWN+1,X         RELEASE RATE
Ø8FB  A9 ØØ     129Ø          LDA #Ø              DL:=Ø
Ø8FD  9D BC ØA  13ØØ          STA DESIRE,X        CURRENT SUSTAIN LEVEL:=Ø
Ø9ØØ  9D BD ØA  131Ø          STA DESIRE+1,X
Ø9Ø3  9D BE ØA  132Ø          STA CURSUS,X
Ø9Ø6  9D BF ØA  133Ø          STA CURSUS+1,X
Ø9Ø9  A9 FF     134Ø  DECR    LDA #$FF          DECREMENT TIME REMAINING
Ø9ØB  DE A6 ØA  135Ø          DEC TIME,X
Ø9ØE  DD A6 ØA  136Ø          CMP TIME,X
Ø911  DØ Ø8     137Ø          BNE NEXTL
Ø913  DE A7 ØA  138Ø          DEC TIME+1,X
Ø916  DD A7 ØA  139Ø          CMP TIME+1,X
Ø919  FØ 15     14ØØ          BEQ PROCES        BRANCH IF NO TIME LEFT
Ø91B  8A        141Ø  NEXTL   TXA               CONTINUE WITH
Ø91C  18        142Ø          CLC                NEXT PART
Ø91D  69 1A     143Ø          ADC #ASIZE
Ø91F  AA        144Ø          TAX
Ø92Ø  E6 Ø2     145Ø          INC /COUNT
Ø922  A5 Ø2     146Ø          LDA /COUNT
Ø924  C5 Ø7     147Ø          CMP /PARNUM
Ø926  DØ B7     148Ø          BNE LENGTH
Ø928  2C 64 CØ  149Ø  WAIT    BIT $CØ64         WAIT FOR TIMER
Ø92B  3Ø FB     15ØØ          BMI WAIT
Ø92D  4C 3C Ø8  151Ø          JMP MAIN
Ø93Ø            152Ø  * SONG DATA COMMAND PROCESSING SECTION
Ø93Ø  8A        153Ø  PROCES  TXA
Ø931  A8        154Ø          TAY
Ø932  A5 Ø2     155Ø          LDA /COUNT
Ø934  ØA        156Ø          ASL A
Ø935  AA        157Ø          TAX
Ø936  A1 Ø8     158Ø          LDA (PARPNT,X)    GET COMMAND TYPE
Ø938  C9 CB     159Ø          CMP #2Ø3
Ø93A  DØ Ø4     16ØØ          BNE NOSTOP        BRANCH IF NOT "STOP"
Ø93C            161Ø  * PROCESS STOP COMMAND
Ø93C  98        162Ø          TYA               DO NOTHING
Ø93D  AA        163Ø          TAX
Ø93E  BØ DB     164Ø          BCS NEXTL
Ø94Ø  F6 Ø8     165Ø  NOSTOP  INC /PARPNT,X
Ø942  DØ Ø2     166Ø          BNE NOCAR1
Ø944  F6 Ø9     167Ø          INC /PARPNT+1,X
Ø946  C9 CØ     168Ø  NOCAR1  CMP #192
Ø948  BØ 57     169Ø          BCS NPITCH        BRANCH IF NOT "PITCH"
Ø94A            17ØØ  * PROCESS PITCH COMMAND
Ø94A  39 AB ØA  171Ø          AND TRANS+1,Y
Ø94D  79 AA ØA  172Ø          ADC TRANS,Y
Ø95Ø  86 Ø4     173Ø          STX /TEMP2        COMPUTE DIVISOR
Ø952  A2 ØØ     174Ø          LDX #Ø            DIVIDE PITCH BY 24
Ø954  C9 18     175Ø  DIV     CMP #24
Ø956  9Ø Ø5     176Ø          BCC DIV1
Ø958  E9 18     177Ø          SBC #24
Ø95A  E8        178Ø          INX
Ø95B  DØ F7     179Ø          BNE DIV
Ø95D  86 Ø3     18ØØ  DIV1    STX /TEMP1
```

```
Ø95F  ØA         181Ø          ASL A               LOOK UP
Ø96Ø  AA         182Ø          TAX                  SUB-OCTAVE DIVISOR
Ø961  BD 76 ØA   183Ø          LDA TABLE,X
Ø964  85 Ø5      184Ø          STA /TEMP3
Ø966  BD 77 ØA   185Ø          LDA TABLE+1,X
Ø969  85 Ø6      186Ø          STA /TEMP3+1
Ø96B  A6 Ø3      187Ø          LDX /TEMP1
Ø96D  CA         188Ø OCTAVE   DEX                 DIVIDE DIVISOR
Ø96E  3Ø Ø7      189Ø          BMI ROUND            TO RIGHT OCTAVE
Ø97Ø  46 Ø6      19ØØ          LSR /TEMP3+1
Ø972  66 Ø5      191Ø          ROR /TEMP3
Ø974  4C 6D Ø9   192Ø          JMP OCTAVE
Ø977  9Ø Ø6      193Ø ROUND    BCC SENDP           ROUND RESULT
Ø979  E6 Ø5      194Ø          INC /TEMP3
Ø97B  DØ Ø2      195Ø          BNE SENDP
Ø97D  E6 Ø6      196Ø          INC /TEMP3+1
Ø97F  BE B6 ØA   197Ø SENDP    LDX CHAN,Y          SEND PITCH TO UNIT
Ø982  A5 Ø5      198Ø          LDA /TEMP3
Ø984  9D 84 CØ   199Ø          STA $CØ84,X
Ø987  A5 Ø6      2ØØØ          LDA /TEMP3+1
Ø989  9D 84 CØ   2Ø1Ø          STA $CØ84,X
Ø98C  A2 Ø6      2Ø2Ø          LDX #6              START "ADSR" CYCLE
Ø98E  84 Ø3      2Ø3Ø          STY /TEMP1
Ø99Ø  B9 AE ØA   2Ø4Ø CYCLE    LDA DECAY,Y
Ø993  99 BA ØA   2Ø5Ø          STA DOWN,Y
Ø996  C8         2Ø6Ø          INY
Ø997  CA         2Ø7Ø          DEX
Ø998  DØ F6      2Ø8Ø          BNE CYCLE
Ø99A  A6 Ø4      2Ø9Ø          LDX /TEMP2
Ø99C  A4 Ø3      21ØØ          LDY /TEMP1          STORE NOTE TIME
Ø99E  4C C3 Ø9   211Ø          JMP STORD1
Ø9A1  DØ 3B      212Ø NPITCH   BNE NREST           BRANCH IF NOT "REST"
Ø9A3             213Ø * PROCESS REST COMMAND
Ø9A3  B9 B4 ØA   214Ø          LDA RELEAS,Y        DO A "RELEASE"
Ø9A6  99 BA ØA   215Ø          STA DOWN,Y
Ø9A9  B9 B5 ØA   216Ø          LDA RELEAS+1,Y
Ø9AC  99 BB ØA   217Ø          STA DOWN+1,Y
Ø9AF  A9 ØØ      218Ø          LDA #Ø
Ø9B1  99 BC ØA   219Ø          STA DESIRE,Y
Ø9B4  99 BD ØA   22ØØ          STA DESIRE+1,Y
Ø9B7  99 BE ØA   221Ø          STA CURSUS,Y
Ø9BA  99 BF ØA   222Ø          STA CURSUS+1,Y
Ø9BD  18         223Ø          CLC
Ø9BE  84 Ø3      224Ø STORD    STY /TEMP1          STORE PARAMETER
Ø9CØ  65 Ø3      225Ø          ADC /TEMP1           IN PARAMETER AREA
Ø9C2  A8         226Ø          TAY
Ø9C3  A1 Ø8      227Ø STORD1   LDA (PARPNT,X)
Ø9C5  99 A6 ØA   228Ø          STA TIME,Y
Ø9C8  F6 Ø8      229Ø          INC /PARPNT,X
Ø9CA  DØ Ø2      23ØØ          BNE NOCAR2
Ø9CC  F6 Ø9      231Ø          INC /PARPNT+1,X
Ø9CE  A1 Ø8      232Ø NOCAR2   LDA (PARPNT,X)
Ø9DØ  99 A7 ØA   233Ø          STA TIME+1,Y
Ø9D3  F6 Ø8      234Ø FIXUP    INC /PARPNT,X
Ø9D5  DØ Ø2      235Ø          BNE NOCAR3
Ø9D7  F6 Ø9      236Ø          INC /PARPNT+1,X
Ø9D9  A6 Ø3      237Ø NOCAR3   LDX /TEMP1
Ø9DB  4C Ø9 Ø9   238Ø          JMP DECR
Ø9DE  85 Ø3      239Ø NREST    STA /TEMP1          SET SO COMMAND
Ø9EØ  A9 ØØ      24ØØ          LDA #Ø               TAKES ZERO TIME
Ø9E2  99 A6 ØA   241Ø          STA TIME,Y
```

```
Ø9E5  99 A7 ØA   242Ø          STA TIME+1,Y
Ø9E8  A5 Ø3      243Ø          LDA /TEMP1
Ø9EA  C9 C9      244Ø          CMP #2Ø1
Ø9EC  BØ Ø5      245Ø          BCS NSTORE           BRANCH IF NOT A
Ø9EE  E9 BF      246Ø          SBC #191              STORED COMMAND
Ø9FØ  ØA         247Ø          ASL A
Ø9F1  DØ CB      248Ø          BNE STORD
Ø9F3  84 Ø3      249Ø NSTORE   STY /TEMP1
Ø9F5  DØ 34      25ØØ          BNE NOCALL           BRANCH IF NOT A "CALL"
Ø9F7             251Ø * PROCESS CALL COMMAND
Ø9F7  A1 Ø8      252Ø          LDA (PARPNT,X)       COMPUTE CALLED ADDRESS
Ø9F9  18         253Ø          CLC
Ø9FA  65 ØØ      254Ø          ADC /SPNTR
Ø9FC  85 Ø5      255Ø          STA /TEMP3
Ø9FE  F6 Ø8      256Ø          INC /PARPNT,X
ØAØØ  DØ Ø2      257Ø          BNE NOCAR4
ØAØ2  F6 Ø9      258Ø          INC /PARPNT+1,X
ØAØ4  A1 Ø8      259Ø NOCAR4   LDA (PARPNT,X)
ØAØ6  65 Ø1      26ØØ          ADC /SPNTR+1
ØAØ8  85 Ø6      261Ø          STA /TEMP3+1
ØAØA  8A         262Ø          TXA                  STORE RETURN ADDRESS
ØAØB  A8         263Ø          TAY
ØAØC  B5 Ø8      264Ø          LDA /PARPNT,X
ØAØE  69 Ø1      265Ø          ADC #1
ØA1Ø  91 Ø5      266Ø          STA (TEMP3),Y
ØA12  C8         267Ø          INY
ØA13  B5 Ø9      268Ø          LDA /PARPNT+1,X
ØAI5  69 ØØ      269Ø          ADC #Ø
ØA17  91 Ø5      27ØØ          STA (TEMP3),Y
ØA19  A5 Ø7      271Ø          LDA /PARNUM          ADVANCE CALLING
ØA1B  ØA         272Ø          ASL A                 ADDRESS OVER
ØA1C  65 Ø5      273Ø          ADC /TEMP3             RETURN ADDRESSES
ØA1E  95 Ø8      274Ø          STA /PARPNT,X
ØA2Ø  A5 Ø6      275Ø          LDA /TEMP3+1
ØA22  69 ØØ      276Ø          ADC #Ø
ØA24  95 Ø9      277Ø          STA /PARPNT+1,X
ØA26  A6 Ø3      278Ø          LDX /TEMP1
ØA28  4C Ø9 Ø9   279Ø          JMP DECR
ØA2B  C9 CC      28ØØ NOCALL   CMP #2Ø4
ØA2D  BØ 22      281Ø          BCS NORET            BRANCH IF NOT "RETURN"
ØA2F             282Ø * PROCESS RETURN COMMAND
ØA2F  A1 Ø8      283Ø          LDA (PARPNT,X)       COMPUTE RETURN ADDRESS
ØA31  65 ØØ      284Ø          ADC /SPNTR            ADDRESS
ØA33  85 Ø5      285Ø          STA /TEMP3
ØA35  F6 Ø8      286Ø          INC /PARPNT,X
ØA37  DØ Ø2      287Ø          BNE NOCAR5
ØA39  F6 Ø9      288Ø          INC /PARPNT+1,X
ØA3B  A1 Ø8      289Ø NOCAR5   LDA (PARPNT,X)
ØA3D  65 Ø1      29ØØ          ADC /SPNTR+1
ØA3F  85 Ø6      291Ø          STA /TEMP3+1
ØA41  8A         292Ø          TXA                  GO TO
ØA42  A8         293Ø          TAY                   RETURN ADDRESS
ØA43  B1 Ø5      294Ø          LDA (TEMP3),Y
ØA45  95 Ø8      295Ø          STA /PARPNT,X
ØA47  C8         296Ø          INY
ØA48  B1 Ø5      297Ø          LDA (TEMP3),Y
ØA4A  95 Ø9      298Ø          STA /PARPNT+1,X
ØA4C  A6 Ø3      299Ø          LDX /TEMP1
ØA4E  4C Ø9 Ø9   3ØØØ          JMP DECR
ØA51  DØ 13      3Ø1Ø NORET    BNE NOTMPO           BRANCH IF NOT "TEMPO"
ØA53             3Ø2Ø * PROCESS TEMPO COMMAND
```

```
ØA53  A1 Ø8      3Ø3Ø          LDA (PARPNT,X)
ØA55  8D 3F Ø8   3Ø4Ø          STA PLACEI+I
ØA58  F6 Ø8      3Ø5Ø          INC /PARPNT,X
ØA5A  DØ Ø2      3Ø6Ø          BNE NOCAR7
ØA5C  F6 Ø9      3Ø7Ø          INC /PARPNT+1,X
ØA5E  A1 Ø8      3Ø8Ø NOCAR7   LDA (PARPNT,X)
ØA6Ø  8D 44 Ø8   3Ø9Ø          STA PLACE2+1
ØA63  4C D3 Ø9   31ØØ          JMP FIXUP
ØA66  C9 FE      311Ø NOTMPO   CMP #254
ØA68  BØ Ø9      312Ø          BCS END          BRANCH IF NOT A "NOP"
ØA6A  F6 Ø8      313Ø          INC /PARPNT,X
ØA6C  DØ Ø2      314Ø          BNE NOCAR6
ØA6E  F6 Ø9      315Ø          INC /PARPNT+I,X
ØA7Ø  4C D3 Ø9   316Ø NOCAR6   JMP FIXUP
ØA73  68         317Ø END      PLA              "PROCESS" END COMMAND
ØA74  AA         3I8Ø          TAX              RESTORE X
ØA75  6Ø         319Ø          RTS               AND RETURN
ØA76             32ØØ * SUB-OCTAVE DIVISOR TABLE
ØA76  2Ø FD      321Ø TABLE    DEF 648ØØ
ØA78  EB F5      322Ø          DEF 62955
ØA7A  EB EE      323Ø          DEF 6I163
ØA7C  1E E8      324Ø          DEF 59422
ØA7E  82 E1      325Ø          DEF 5773Ø
ØA8Ø  17 DB      326Ø          DEF 56Ø87
ØA82  DA D4      327Ø          DEF 5449Ø
ØA84  CB CE      328Ø          DEF 52939
ØA86  E8 C8      329Ø          DEF 5I432
ØA88  3Ø C3      33ØØ          DEF 49968
ØA8A  A1 BD      331Ø          DEF 48545
ØA8C  3B B8      332Ø          DEF 47163
ØA8E  FD B2      333Ø          DEF 45821
ØA9Ø  E4 AD      334Ø          DEF 44516
ØA92  F1 A8      335Ø          DEF 43249
ØA94  22 A4      336Ø          DEF 42Ø18
ØA96  75 9F      337Ø          DEF 4Ø821
ØA98  EB 9A      338Ø          DEF 39659
ØA9A  82 96      339Ø          DEF 3853Ø
ØA9C  39 92      34ØØ          DEF 37433
ØA9E  1Ø 8E      341Ø          DEF 36368
ØAAØ  Ø4 8A      342Ø          DEF 35332
ØAA2  17 86      343Ø          DEF 34327
ØAA4  45 82      344Ø          DEF 33349
ØAA6             345Ø * COMMAND PARAMETER AREA
ØAA6             346Ø TIME     BSS 2            TIME REMAINING
ØAA8             347Ø GAP      BSS 2            GAP SIZE
ØAAA             348Ø TRANS    BSS 2            TRANSPOSE VALUE
ØAAC             349Ø ATTACK   BSS 2            ATTACK RATE
ØAAE             35ØØ DECAY    BSS 6            DECAY RATE
ØABØ             35IØ VOLUME   EQU DECAY+2      VOLUME LEVEL
ØAB2             352Ø SUSTAN   EQU VOLUME+2     SUSTAIN LEVEL
ØAB4             353Ø RELEAS   BSS 2            RELEASE RATE
ØAB6             354Ø CHAN     BSS 2            CHANNEL NUMBER
ØAB8             355Ø LOUDNS   BSS 2            CURRENT LOUDNESS
ØABA             356Ø DOWN     BSS 6            CURRENT DECAY RATE
ØABC             357Ø DESIRE   EQU DOWN+2       DESIRED LOUDNESS
ØABE             358Ø CURSUS   EQU DESIRE+2     CURRENT SUSTAIN LEVEL
ØØ1A             359Ø ASIZE    EQU *-TIME       PARAMETER AREA SIZE
ØACØ             36ØØ          BSS ASIZE*8      OTHER 8 PARTS
ØB9Ø             36IØ          END
```

## CHROMA (INTEGER VERSION)

```
ØØØØ              * CHROMA SUBROUTINE
ØØØØ              *
ØØØØ              * BY JOHN RIDGES
ØØØØ              *
ØØØØ              * ALF PRODUCTS INC.
ØØØØ              *
2ØØØ                        ORG $2ØØØ
2ØØØ              * INTEGER BASIC LINE HEADER
2ØØØ  AC                    DAT LINE2-*
2ØØ1  FF FF                 DEF $FFFF
2ØØ3  5D                    DAT $5D
2ØØ4              * PARAMETERS TO SUBROUTINES
2ØØ4  ØØ          PITCH     DAT Ø                FREQUENCY IN QUARTER STEPS
2ØØ5  ØØ          CHAN      DAT Ø                CHANNEL TO BE PROGRAMMED
2ØØ6  ØØ          SLOT      DAT Ø                SLOT OF UNIT TIMES 16
2ØØ7  ØØ          OFFSET    DAT Ø                QUARTER STEP OFFSET
2ØØ8  ØØ          WIDTH     DAT Ø                VARIABLE PULSE WIDTH
2ØØ9  ØØ          DIVSRL    DAT Ø                RESULT DIVISOR
2ØØA  ØØ          DIVSRH    DAT Ø
2ØØB              * ENTRY POINT FOR CHROMA SUBROUTINE
2ØØB  18                    CLC
2ØØC  9Ø 33                 BCC ENTRY
2ØØE              * ENTRY POINT FOR PULSE SUBROUTINE
2ØØE  38                    SEC
2ØØF  BØ 3Ø                 BCS ENTRY
2Ø11              * QUARTER TONE DIVISOR TABLE
2Ø11  2Ø FD       TABLE     DEF 648ØØ
2Ø13  EB F5                 DEF 62955
2Ø15  EB EE                 DEF 61163
2Ø17  1E E8                 DEF 59422
2Ø19  82 E1                 DEF 5773Ø
2Ø1B  17 DB                 DEF 56Ø87
2Ø1D  DA D4                 DEF 5449Ø
2Ø1F  CB CE                 DEF 52939
2Ø21  E8 C8                 DEF 51432
2Ø23  3Ø C3                 DEF 49968
2Ø25  A1 BD                 DEF 48545
2Ø27  3B B8                 DEF 47163
2Ø29  FD B2                 DEF 45821
2Ø2B  E4 AD                 DEF 44516
2Ø2D  F1 A8                 DEF 43249
2Ø2F  22 A4                 DEF 42Ø18
2Ø31  75 9F                 DEF 4Ø821
2Ø33  EB 9A                 DEF 39659
2Ø35  82 96                 DEF 3853Ø
2Ø37  39 92                 DEF 37433
2Ø39  1Ø 8E                 DEF 36368
2Ø3B  Ø4 8A                 DEF 35332
2Ø3D  17 86                 DEF 34327
2Ø3F  45 82                 DEF 33349
2Ø41              * SET UP BASE ADDRESS FOR SUBROUTINES
2Ø41  A9 ØØ       ENTRY     LDA #Ø               SET TO LOW BYTE OF HIMEM
2Ø43  85 54                 STA /AUXL
2Ø45  A9 ØØ                 LDA #Ø               SET TO HIGH BYTE OF HIMEM-2
2Ø47  85 55                 STA /AUXH
2Ø49  8A                    TXA                  SAVE THE X REGISTER
2Ø4A  48                    PHA
2Ø4B  9Ø 4B                 BCC CHROMA           EXECUTE DESIRED SUBROUTINE
```

```
2Ø4D              * COMPUTE XTNDL.ACH:=(DIVSR*(WIDTH+I))/256
2Ø4D  AØ B4       PULSE    LDY #REF+WIDTH
2Ø4F  A2 FE                LDX #-2           STORE WIDTH IN XTNDL
2Ø51  B1 54       PULSEØ   LDA (AUXL),Y        AND DIVSRL IN XTNDH
2Ø53  95 54                STA /XTNDL+2,X
2Ø55  C8                   INY
2Ø56  E8                   INX
2Ø57  DØ F8                BNE PULSEØ
2Ø59  B1 54                LDA (AUXL),Y      STORE DIVSRH IN ACL
2Ø5B  85 5Ø                STA /ACL
2Ø5D  86 51                STX /ACH          CLEAR ACH
2Ø5F  A2 Ø8                LDX #8            XTNDL.ACH.ACL:=(DIVSR*XTNDL)
2Ø61  Ø6 53       PULSE1   ASL /XTNDH         +ACH.ACL.XTNDH
2Ø63  26 5Ø                ROL /ACL
2Ø65  26 51                ROL /ACH
2Ø67  26 52                ROL /XTNDL
2Ø69  9Ø 13                BCC PULSE2
2Ø6B  18                   CLC
2Ø6C  88                   DEY
2Ø6D  B1 54                LDA (AUXL),Y
2Ø6F  65 5Ø                ADC /ACL
2Ø71  85 5Ø                STA /ACL
2Ø73  C8                   INY
2Ø74  B1 54                LDA (AUXL),Y
2Ø76  65 51                ADC /ACH
2Ø78  85 51                STA /ACH
2Ø7A  9Ø Ø2                BCC PULSE2
2Ø7C  E6 52                INC /XTNDL
2Ø7E  CA          PULSE2   DEX
2Ø7F  DØ EØ                BNE PULSEI
2Ø81  AØ B1                LDY #REF+CHAN
2Ø83  B1 54                LDA (AUXL),Y
2Ø85  3Ø ØE                BMI PULSE3        BRANCH IF NO-SEND FLAG SET
2Ø87  C8                   INY               OR IN SLOT TO
2Ø88  11 54                ORA (AUXL),Y        FORM UNIT ADDRESS
2Ø8A  AA                   TAX
2Ø8B  A5 51                LDA /ACH          SEND XTNDL.ACH TO UNIT
2Ø8D  9D 84 CØ             STA $CØ84,X
2Ø9Ø  A5 52                LDA /XTNDL
2Ø92  9D 84 CØ             STA $CØ84,X
2Ø95  68          PULSL3   PLA               RESTORE X AND RETURN
2Ø96  AA                   TAX
2Ø97  6Ø                   RTS
2Ø98  AØ BØ       CHROMA   LDY #REF+PITCH
2Ø9A  B1 54                LDA (AUXL),Y      DIVIDE PITCH BY 24 TO GET
2Ø9C  A2 ØØ                LDX #Ø              A:=SUBOCTAVE
2Ø9E  C9 18       CHROM1   CMP #24             X:=OCTAVE
2ØAØ  9Ø ØE                BCC CHROM2
2ØA2  E9 18                SBC #24
2ØA4  E8                   INX
2ØA5  DØ F7                BNE CHROM1
2ØA7              * LINE ONE TRAILER
2ØA7  5D 5D                DEF $5D5D
2ØA9  5D 5D                DEF $5D5D
2ØAB  Ø1                   DAT 1
2ØAC              * LINE 2 HEADER
2ØAC  A8          LINE2    DAT LINE3-*
2ØAD  FF FF                DEF $FFFF
2ØAF  5D                   DAT $5D
2ØBØ              * BACK TO CHROMA
2ØBØ  ØA          CHROM2   ASL A             GET THE PROPER DIVISOR
```

```
2ØB1  69 BE     12ØØ              ADC #REF+TABLE+1   FROM THE TABLE
2ØB3  A8        121Ø              TAY
2ØB4  B1 54     122Ø              LDA (AUXL),Y
2ØB6  85 51     123Ø              STA /ACH
2ØB8  88        124Ø              DEY
2ØB9  B1 54     125Ø              LDA (AUXL),Y
2ØBB  CA        126Ø              DEX                DIVISOR:=DIVISOR/(2↑OCTAVE)
2ØBC  3Ø Ø6     127Ø              BMI CHROM4
2ØBE  46 51     128Ø    CHROM3    LSR /ACH
2ØCØ  6A        129Ø              ROR A
2ØC1  CA        13ØØ              DEX
2ØC2  1Ø FA     131Ø              BPL CHROM3
2ØC4  69 ØØ     132Ø    CHROM4    ADC #Ø             ROUND THE RESULT
2ØC6  9Ø Ø2     133Ø              BCC CHROM5
2ØC8  E6 51     134Ø              INC /ACH
2ØCA  AØ B5     135Ø    CHROM5    LDY #REF+DIVSRL
2ØCC  91 54     136Ø              STA (AUXL),Y       STORE THE RESULT
2ØCE  C8        137Ø              INY                  IN DIVSR
2ØCF  A5 51     138Ø              LDA /ACH
2ØD1  91 54     139Ø              STA (AUXL),Y
2ØD3  AØ B3     14ØØ              LDY #REF+OFFSET
2ØD5  B1 54     141Ø              LDA (AUXL),Y
2ØD7  FØ 5A     142Ø              BEQ CHROMØ         BRANCH IF NO OFFSET
2ØD9            143Ø    * COMPUTE DIVSR:=DIVSR-(DIVSR*OFFSET)/8993
2ØD9  85 52     144Ø              STA /XTNDL         SAVE OFFSET
2ØDB  E8        145Ø              INX                CLEAR XTNDH, ACL, AND ACH
2ØDC  86 53     146Ø              STX /XTNDH
2ØDE  86 5Ø     147Ø              STX /ACL
2ØEØ  86 51     148Ø              STX /ACH
2ØE2  A2 Ø8     149Ø              LDX #8             XTNDL.ACH.ACL:=(DIVSR*XTNDL)
2ØE4  AØ B5     15ØØ              LDY #REF+DIVSRL     +ACH.ACL*256
2ØE6  Ø6 5Ø     151Ø    CHROM6    ASL /ACL
2ØE8  26 51     152Ø              ROL /ACH
2ØEA  26 52     153Ø              ROL /XTNDL
2ØEC  9Ø 13     154Ø              BCC CHROM7
2ØEE  18        155Ø              CLC
2ØEF  B1 54     156Ø              LDA (AUXL),Y
2ØF1  65 5Ø     157Ø              ADC /ACL
2ØF3  85 5Ø     158Ø              STA /ACL
2ØF5  C8        159Ø              INY
2ØF6  B1 54     16ØØ              LDA (AUXL),Y
2ØF8  65 51     161Ø              ADC /ACH
2ØFA  85 51     162Ø              STA /ACH
2ØFC  88        163Ø              DEY
2ØFD  9Ø Ø2     164Ø              BCC CHROM7
2ØFF  E6 52     165Ø              INC /XTNDL
21Ø1  CA        166Ø    CHROM7    DEX
21Ø2  DØ E2     167Ø              BNE CHROM6
21Ø4  AØ 1Ø     168Ø              LDY #16            AC:=XTND.AC/8993
21Ø6  Ø6 5Ø     169Ø    CHROM8    ASL /ACL           XTND:=XTND.AC MOD 8993
21Ø8  26 51     17ØØ              ROL /ACH
21ØA  26 52     171Ø              ROL /XTNDL
21ØC  26 53     172Ø              ROL /XTNDH
21ØE  38        173Ø              SEC
21ØF  A5 52     174Ø              LDA /XTNDL
2111  E9 21     175Ø              SBC #33
2113  AA        176Ø              TAX
2114  A5 53     177Ø              LDA /XTNDH
2116  E9 23     178Ø              SBC #35
2118  9Ø Ø6     179Ø              BCC CHROM9
211A  86 52     18ØØ              STX /XTNDL
```

```
211C  85 53     181Ø            STA /XTNDH
211E  E6 5Ø     182Ø            INC /ACL
212Ø  88        183Ø  CHROM9    DEY
2121  DØ E3     184Ø            BNE CHROM8
2123  AØ B5     185Ø            LDY #REF+DIVSRL
2125  B1 54     186Ø            LDA (AUXL),Y        DIVSR:=DIVSR-AC
2127  38        187Ø            SEC
2128  E5 5Ø     188Ø            SBC /ACL
212A  91 54     189Ø            STA (AUXL),Y
212C  C8        19ØØ            INY
212D  B1 54     191Ø            LDA (AUXL),Y
212F  E5 51     192Ø            SBC /ACH
2131  91 54     193Ø            STA (AUXL),Y
2133  AØ B1     194Ø  CHROMØ    LDY #REF+CHAN
2135  B1 54     195Ø            LDA (AUXL),Y
2137  3Ø 11     196Ø            BMI CHROM:          BRANCH IF NO-SEND FLAG SET
2139  C8        197Ø            INY                 OR IN SLOT TO
213A  11 54     198Ø            ORA (AUXL),Y          FORM UNIT ADDRESS
213C  AA        199Ø            TAX
213D  AØ B5     2ØØØ            LDY #REF+DIVSRL
213F  B1 54     2Ø1Ø            LDA (AUXL),Y        SEND DIVISOR TO UNIT
2141  9D 84 CØ  2Ø2Ø            STA $CØ84,X
2144  C8        2Ø3Ø            INY
2145  B1 54     2Ø4Ø            LDA (AUXL),Y
2147  9D 84 CØ  2Ø5Ø            STA $CØ84,X
214A  68        2Ø6Ø  CHROM:    PLA                 RESTORE X AND RETURN
214B  AA        2Ø7Ø            TAX
214C  6Ø        2Ø8Ø            RTS
214D            2Ø9Ø  * LINE 2 TRAILER
214D  5D 5D     21ØØ            DEF $5D5D
214F  5D        211Ø            DAT $5D
215Ø  C1 CC     212Ø            DEF $CCC1
2152  C6 Ø1     213Ø            DEF $1C6
2154            214Ø  LINE3     EQU *
EØAC            215Ø  REF       EQU 512-*           HIMEM LOCATION REFERENCE
2154            216Ø  * ON BASE PAGE
ØØ5Ø            217Ø  ACL       EQU $5Ø
ØØ51            218Ø  ACH       EQU $51
ØØ52            219Ø  XTNDL     EQU $52
ØØ53            22ØØ  XTNDH     EQU XTNDL+1
ØØ54            221Ø  AUXL      EQU $54
ØØ55            222Ø  AUXH      EQU AUXL+1
2154            223Ø            END
```

# INDEX

# SIGNAL DESCRIPTIONS

| Pin | Name | Desc. |
|---|---|---|
| 2 | AØ | Address line Ø. 1 LS TTL load. |
| 3 | A1 | " " 1. 1 LS TTL load. |
| 4 | A2 | " " 2. 2 LS TTL loads. |
| 18 | $R/\overline{W}$ | $\text{Read}/\overline{\text{Write}}$. 1 LS TTL load. |
| 23 | INT OUT | Connected to pin 28. |
| 24 | DMA OUT | Connected to pin 27. |
| 25 | +5V | +5 volts, ± 5%. 13Ø mA typical, 215 mA max. |
| 26 | GND | Signal ground. |
| 27 | DMA IN | Connected to pin 24. |
| 28 | INT IN | Connected to pin 23. |
| 33 | -12V | -18 volts to -1Ø.8 volts. 2Ø mA typical, 3Ø mA max. |
| 41 | $\overline{\text{DEV SEL}}$ | Board enable. 2 LS TTL loads. |
| 42 | D7 | Data bus bit 7. 1 LS TTL load. |
| 43 | D6 | " " " 6. 1 LS TTL load. |
| 44 | D5 | " " " 5. 1 LS TTL load. |
| 45 | D4 | " " " 4. 1 LS TTL load. |
| 46 | D3 | " " " 3. 1 LS TTL load. |
| 47 | D2 | " " " 2. 1 LS TTL load. |
| 48 | D1 | " " " 1. 1 LS TTL load. |
| 49 | DØ | " " " Ø. 1 LS TTL load. |
| 5Ø | +12V | +1Ø volts to +18 volts. 25 mA typical, 35 mA max. |

Supply voltages (+5V, -12V, and +12V) should be regulated.

## A4 ACCESS SOCKET

| Pin | Name | Desc. |
|---|---|---|
| 1 | +12 | Connected to +12 volts. |
| 2 | NC | |
| 3 | OUTØ | TTL output of channel Ø. Drives 3 LS loads. |
| 4 | OUT1 | " " " " 1. Drives 3 LS loads. |
| 5 | OUT2 | " " " " 2. Drives 1 LS load. |
| 6 | NC | |
| 7 | -12 | Connected to -12 volts. |
| 8 | GND | Signal ground. |
| 9 | NC | |
| 1Ø | NC | |
| 11 | AUD | Audio out. Source/sink 6.5 mA max. 2.25 to 7.25 volts. |
| 12 | NC | |
| 13 | NC | |
| 14 | +5 | Connected to +5 volts. |

## TTL INPUT REQUIREMENTS

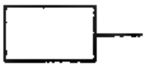
| | MIN | MAX |
|---|---|---|
| High Level Input Voltage | 2 volts | 5.5 volts |
| Low Level Input Voltage | Ø volts | Ø.8 volts |

1 LS load = 2Ø uA at 2.7 volts input and -Ø.4 mA at Ø.4 volts input.

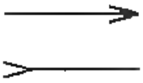
## AUDIO OUTPUTS

Impedance: 7ØØ ohms typical. Output: Ø.91 volts peak.

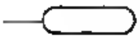
## SCHEMATIC TERMINALS



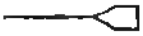
Connection to pin on Apple II Peripheral I/O Bus



Internal connections

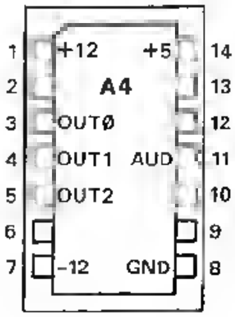


Connection to Audio Out molex pin

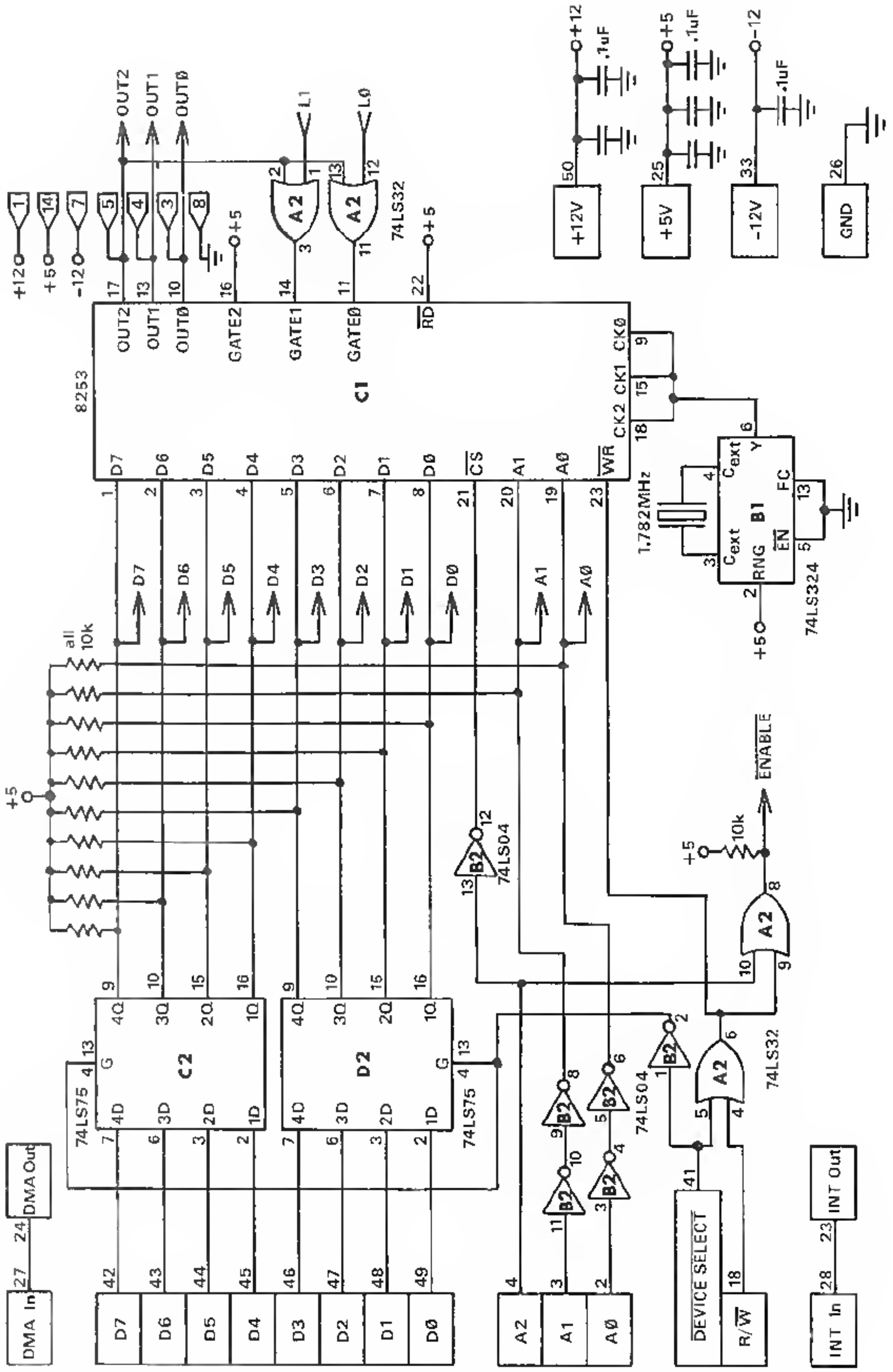


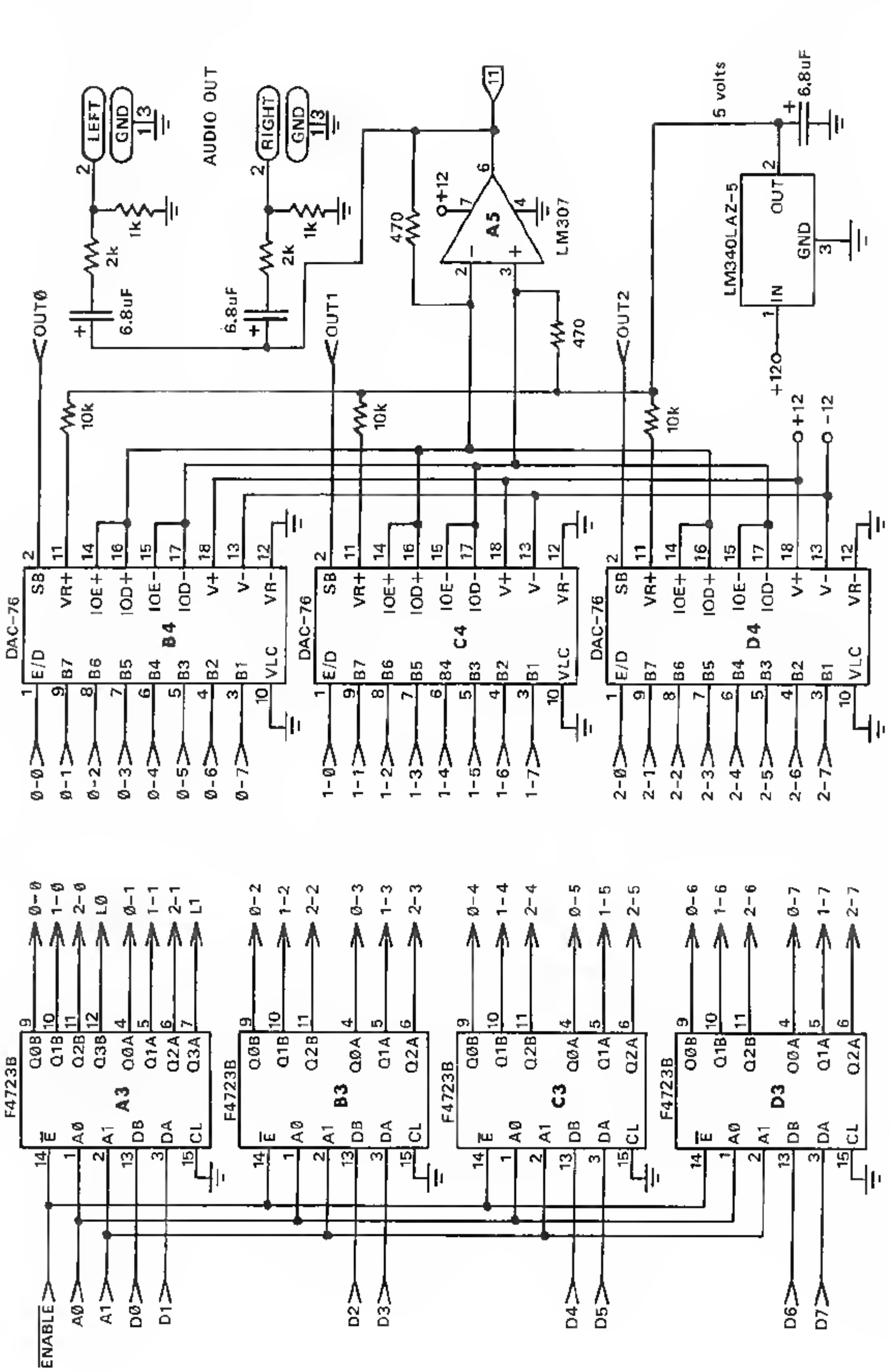
Connection to pin on Access Socket **A4**

**Baldface** characters on schematic (eg. **C2**) refer to component locations. See silkscreen artwork for locations.



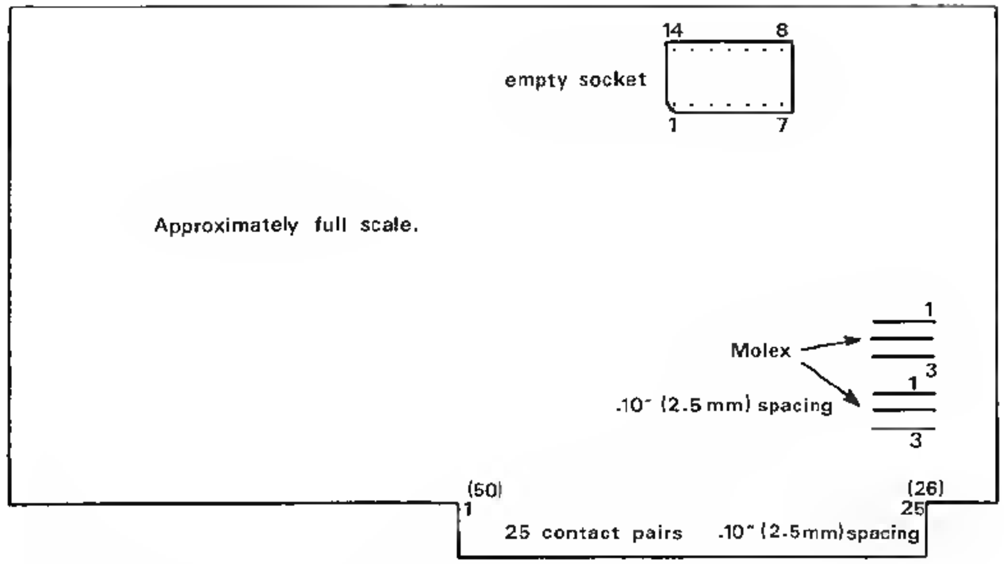
ACCESS SOCKET


8253
C1
OUT2
OUT1
OUTØ
GATE2
GATE1
GATEØ
RD
CKØ
CK1
CK2
D7
D6
D5
D4
D3
D2
D1
DØ
CS
A1
AØ
WR
L1
LØ
A2
74LS32
+5
+12
-12
.1uF
+12V
+5V
-12V
GND
50
25
33
26
1.782MHz
B1
Cext
Y
FC
RNG
EN
74LS324
all 10k
ENABLE
10k
B2
74LS04
C2
D2
74LS75
4Q
3Q
2Q
1Q
4D
3D
2D
1D
G
DMA Out
DMA In
24
27
42
43
44
45
46
47
48
49
A2
A1
AØ
DEVICE SELECT
R/W
41
18
INT Out
INT In
23
28



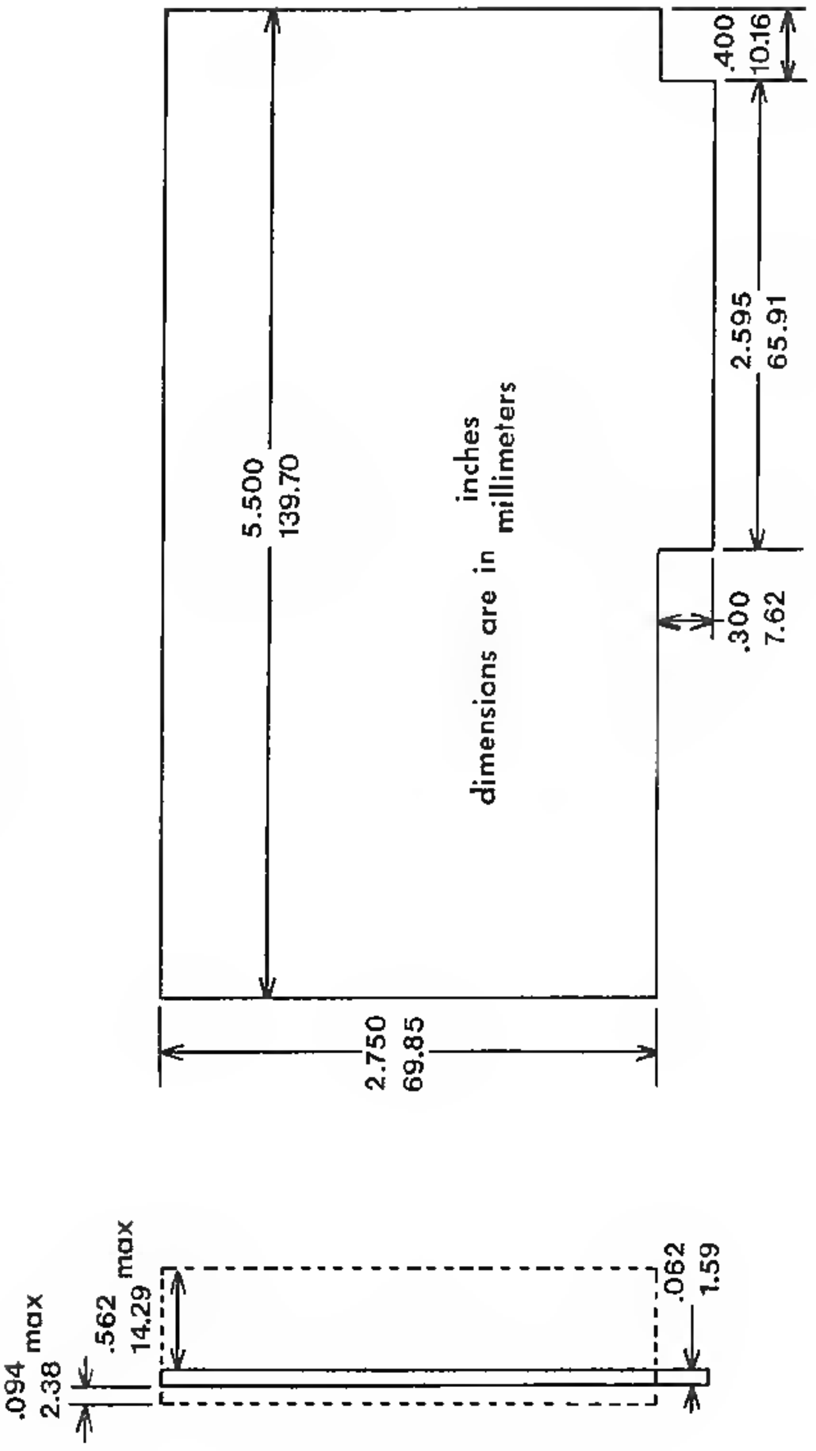
Note: IC's **B4**, **C4**, and **D4** are powered from +12V and −12V; **A5** is powered from +12V; all others are powered from +5V.

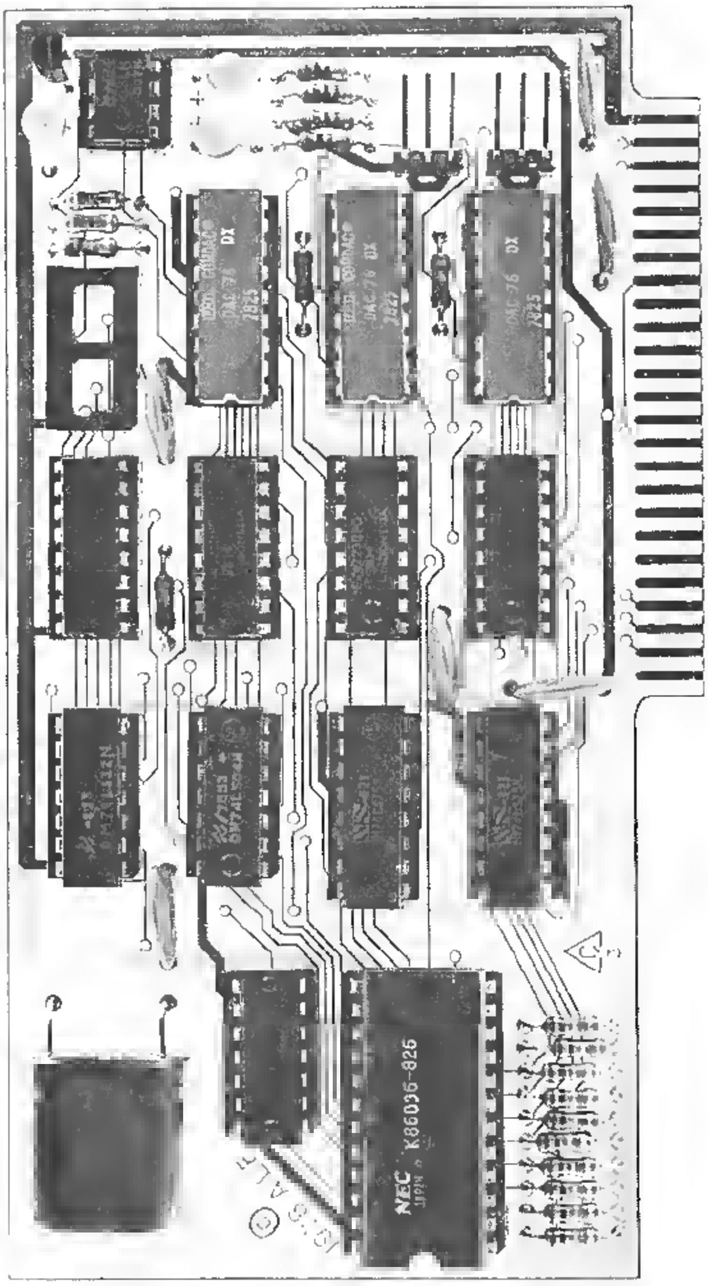
# CONNECTOR LOCATIONS

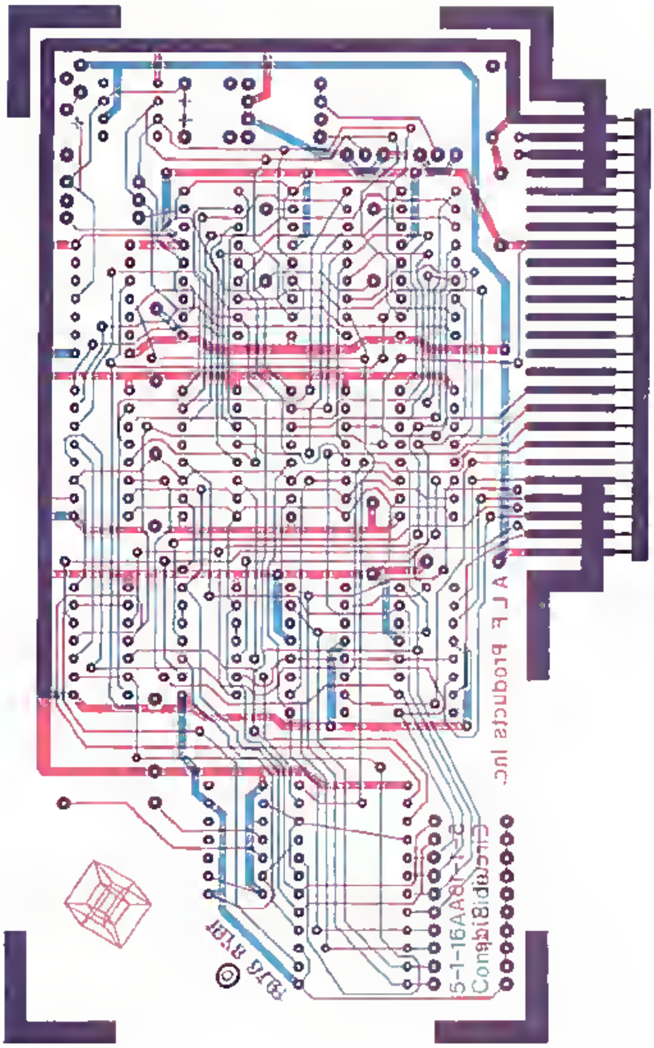


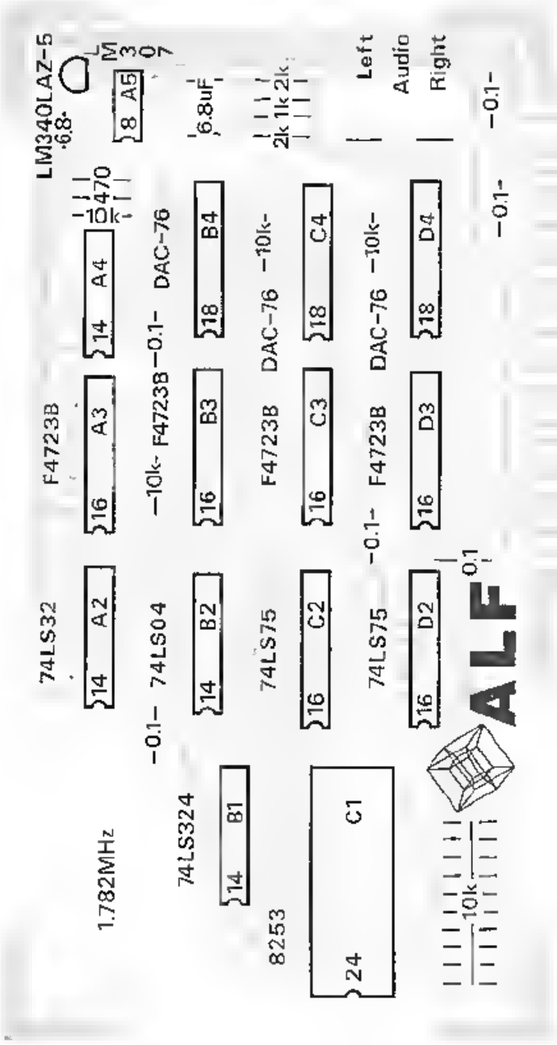
Note: Edge contact mates with Winchester HW25C or equivalent. (Supplied in Apple II.)

# DIMENSIONS


5.500
139.70
dimensions are in inches
millimeters
.400
10.16
2.595
65.91
.300
7.62
2.750
69.85
.094 max
2.38
.562 max
14.29
.062
1.59


DAC-76
DX
7825
NEC
K86036-826


A L F Products Inc.


LM340LAZ-5
6.8
LM307
8 A5
10k
470
14 A4
DAC-76
18 B4
6.8uF
2k 1k 2k
-10k-
18 C4
DAC-76
-10k-
18 D4
Left
Audio
Right
-0.1-
-0.1-
-0.1-
F4723B
16 A3
-10k- F4723B -0.1-
16 B3
F4723B
16 C3
DAC-76
-0.1- F4723B
16 D3
74LS32
14 A2
74LS04
14 B2
74LS75
16 C2
74LS75
16 D2
0.1
ALF
-0.1-
74LS324
14 B1
1.782MHz
8253
24 C1
10k